Cross Layer Performance Improvements in Multi-hop Ad-hoc Networks

---

A thesis submitted for the degree of
Doctor of Philosophy
by
David Murray

Murdoch University
2011

Abstract

The multi hop architecture can provide greater range, spatial efficiency and throughput with lower power consumption than a traditional point coordinated design. In addition, the self-configuring, self-healing characteristics of this architecture enable previously infeasible network scenarios and applications. These benefits have motivated a significant volume of research. Unfortunately, the ISO networking model was not designed for multi hop ad hoc networks and subsequently, some cross layer interactions are necessary to provide adequate performance. This thesis develops and evaluates solutions to multiple performance problems. It makes contributions in the areas of routing, multi-hop contention and acknowledgement efficiency. The results are captured using a real world testbed. The first study evaluates three popular mesh routing protocols. The second contribution creates a novel channel selection mechanism that operates with multiple radios to overcome multi hop performance degradations. The final study shows the extent of the 802.11 acknowledgement overhead. A novel distributed proxy is created to eliminate this overhead in multi hop ad hoc networks.

Acknowledgments

I am most grateful to my supervisors, Mike Dixon and Terry Koziniec. Their open doors and persistent encouragement was invaluable for the completion of this PhD.

I would also like to acknowledge the help of Nik Thompson and say thanks for the countless cups of coffee shared.

Finally, these acknowledgements would not be complete without appreciating the unwavering support of my partner Rosa.

Publications

- David Murray, Michael Dixon and Terry Koziniec, An Experimental Comparison of Routing Protocols in Multi Hop Ad Hoc Networks, *ANTAC: Australasian Telecommunication Networks and Applications Conference*, 2010

This paper performs an experimental comparison of three popular multi hop ad hoc routing protocols. The study found that Babel provided higher throughputs than OLSR or BABEL. This paper was derived from Chapter 2.

- David Murray, Michael Dixon and Terry Koziniec, RDCS: Routing Driven Channel Selection in Multi Radio Ad-hoc Networks, *IWCMC'09: The 5th International Wireless Communications and Mobile Computing Conference*, 2009

This paper proposes, implements and experimentally tests a new channel selection mechanism for multi radio ad hoc nodes called RDCS (Routing Driven Channel Selection). This paper was derived from Chapter 3.

- David Murray, Terry Koziniec and Michael Dixon, Solving Ack Inefficiencies in 802.11 Networks, *IMSAA-09: IEEE International Conference on Internet Multimedia Systems Architecture and Applications*, 2009

This paper proposes a new PEP (Performance Enhancing Proxy) called D-Proxy which is capable of providing reliability at the data link layer. This work shows how using D-Proxy enables 802.11 link layer acknowledgements to be removed, resulting in performance increases. This paper was derived from Chapter 4.

Table of Contents

List of Figures

List of Tables

Chapter I

Introduction

1.1 Technical Advantages of Multi Hop Ad hoc Networks

Multi hop ad hoc networks can offer lower power consumption, greater range, spatial efficiency and throughput. The inverse square rule is a fundamental rule which states that signal strength degrades with the inverse square of the distance. Thus, to double the distance of a wireless transmission, the power must be increased by four times. In Figure 1.1, a battery constrained device with transmission power of 50mW, is providing a transmission distance of 50m. To communicate with another node at a distance of 150m, the transmission power of the radio must be increased to 450mW. As a result, the multi hop architecture can reduce power consumption in battery constrained devices.

In some scenarios, increasing the transmission power may be infeasible due to the power limitations of the radio, battery power or legal wireless limits. Thus, the only option to communicate with a distant device may be to hop the transmission over multiple cooperative nodes. Multi hop ad hoc networks may enable distant nodes to communicate within the limits imposed by the radio, battery power or the law. In Figure 1.2a, two nodes are unable to communicate due to distance. In Figure 1.2b, multi hopping

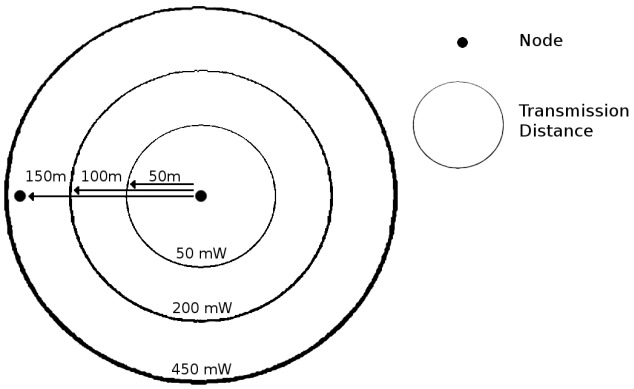


Figure 1.1: Power

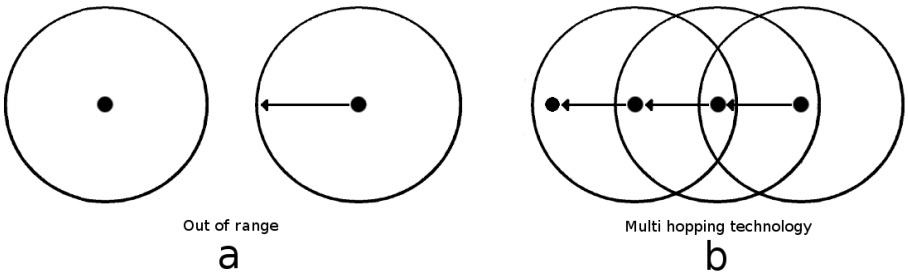


Figure 1.2: Range

technology is enabling communication via two cooperative nodes

Lower power and greater range are not the only benefits provided by the multi hop wireless architecture. In Figure 1.3a, a node is using a large amount of power to communicate with a distant node. In Figure 1.3b, multi hopping technology is being used. The size of the circles indicates the spectral propagation of a particular nodes transmission. Figure 1.3b shows that the multi hop architecture is more spatially efficient than a high powered point coordinated design.

The final advantage; throughput, is offered by a more effective architecture. In the traditional AP-client communication paradigm, shown in Figure 1.4a, communication between two wireless devices occurs via a AP (Access

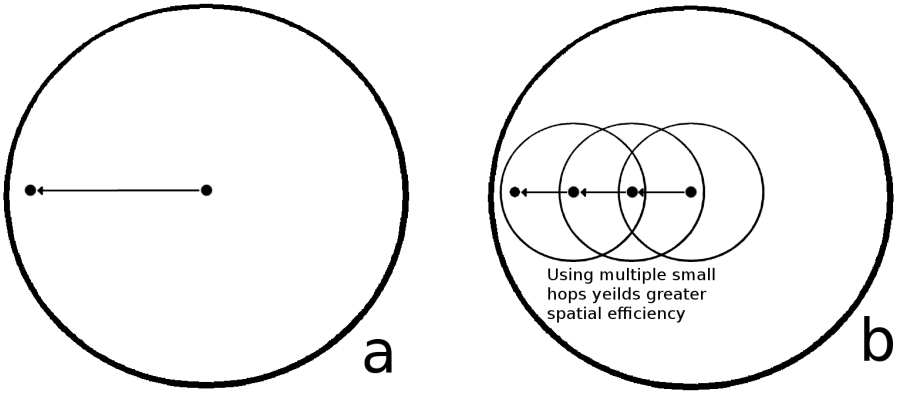


Figure 1.3: Spatial efficiency

Point), which acts as a coordinator for all transmissions. In some scenarios this point coordinated design is wasteful of resources because every packet must be sent twice; once from the sender to the AP and another time from the AP to the receiver. This process of sending a packet twice immediately halves the potential bandwidth of that connection. Furthermore, it is also possible that the sender and receiver are nearby one another but are distant from the AP. In this case, the bit rate of transmissions to the AP may be lower than the achievable bit rate possible from direct communication. In the multi hop ad hoc architecture shown in Figure 1.4b, any node can communicate directly with any other node. There are power, range, spatial efficiency and throughput advantages of the multi hop architecture.

1.2 802.11

Multi hop ad hoc networks are a generic communication architecture being applied to many wireless technologies. There are groups working on

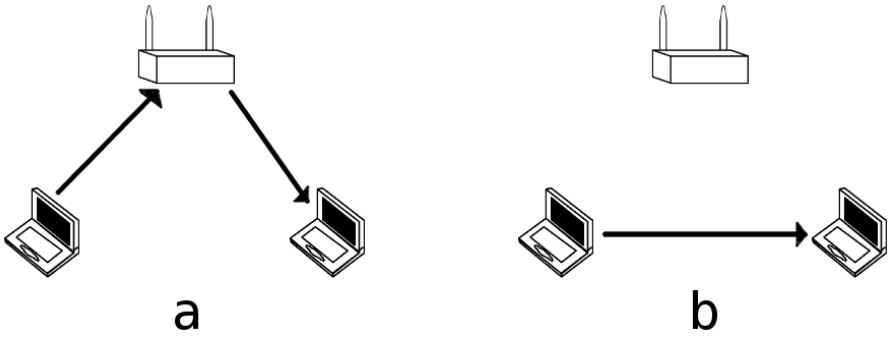


Figure 1.4: Throughput advantages of the Multi hop architecture

multi hop technologies for different wireless systems including WiMAX [1] and Zigbee [2]. In this dissertation, the primary focus is 802.11 [3] or WiFi but the research may be relevant for any wireless technology. 802.11 has a number of important characteristics. Firstly, it is an unlicensed technology. This means that any person can deploy and operate 802.11 networks without spectral licensing fees. Secondly, it is a constantly evolving International IEEE standard which has been in operation for over a decade [3]. The pervasiveness of this open standards technology means that the economies of scale provide a significantly advanced technology at low cost. The IEEE 802.11n Drafts received more comments than any previous IEEE standard; which is indicative of both the importance and magnitude of research interest. In the future, low cost, high bandwidth 802.11 technology is likely to continue to be embedded in greater numbers of electronics which will further enhance the viability of an 802.11 multi hop architecture.

1.3 Terminology and Focus

The terminology used to describe multi hop ad hoc networks is haphazard. Multi hop ad hoc network, wireless mesh networks, and MANET (Mobile Ad hoc Network) are all expressions that have been used interchangeably to describe the same idea. This thesis differentiates these terms based on the degree of mobility, power and type of device. The term MANET, which is the term used by the IETF [4], was intended for mobile wireless ad hoc networks. The original design intentions of MANET, discussed in RFC 2501 [5], describe dynamic topologies and energy constrained operation. In MANETs, the target device is battery powered mobile equipment and thus CPU utilization, power consumption and the ability of the routing protocol to operate with a large number of network changes are key concerns. Another term, wireless mesh networks, also describes a multi hop ad hoc network architecture. However, in mesh networks the type of node is typically a purpose built infrastructure device. These nodes will have limited mobility, higher CPU capabilities, will typically be mains powered, and may be equipped with multiple radios. The use of terminology in this thesis, and the future architecture envisaged is shown in Figure 1.5. In this thesis, mesh networks will be the primary focus, although some discussions may also be applicable for MANETs (Mobile Ad hoc NETworks). Usage of "multi hop ad hoc network" in this thesis, is a generic term used to refer to any multi hopping technology.

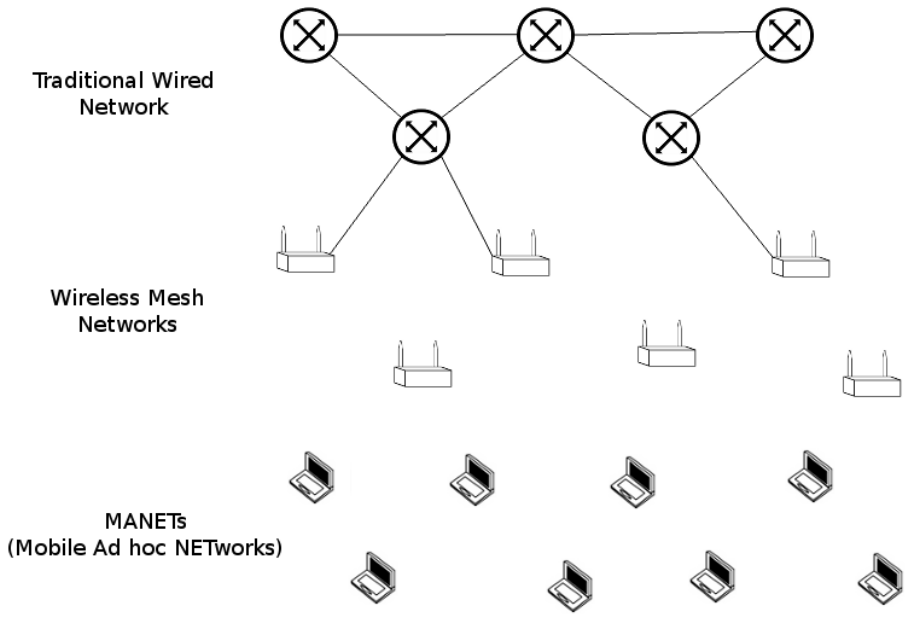


Figure 1.5: Differentiating mesh networks from MANETs

1.4 Mesh Networks

Mesh networks are designed to operate in the presence of lost nodes and links. The self-forming, self-healing characteristics of these networks are designed to enable the network to quickly reform and re-converge [5]. These networks must also operate with minimal planning and administrator control [5]. In addition, they must also be able to operate in the presence of constantly changing link conditions. The dynamic and adaptable characteristics of mesh networks present the opportunity for previously infeasible network scenarios such as:

1. Large municipal and community networks

2. Optimal WLAN deployment: extending access into open areas

3. Developing world communications

4. Disaster recovery and rescue services

1.4.1 Large municipal and community networks

The deployment of traditional 802.11 networks is constrained by 802.11's short transmission range. Using the traditional AP based wireless architecture, many APs, each with an individual wired Ethernet connection to the backbone network are required to provide coverage to large areas. This is the wireless paradox; each AP must be individually wired. With traditional AP based networks, this creates a number of deployment problems because the 100m cabling limitation of copper based Ethernet is often insufficient and the cost of fiber optics is restrictive. The administrative difficulties of laying cables through public and private property are equally problematic. Wireless 802.11 mesh networks can potentially solve this problem by using inexpensive, wireless links between APs; negating the need for a wired backbone to every AP. In recent times, city wide 802.11 mesh networks have been implemented in numerous cities such as Oklahoma City, Philadelphia, and Paris.

1.4.2 Optimal WLAN deployment: extending access into open areas

APs are often unable to be placed in an optimal position because of cabling practicalities. Subsequently, WLANs are most commonly found indoors; in offices, cafes, airports and universities. It is expensive and some-

times impossible to deploy WLANs in outdoor areas. Some historic buildings were never designed for wired infrastructure. These situations could greatly benefit from the freedom wireless mesh networks can provide. Even organizations with existing wired infrastructure may be inclined to use small meshes to position APs in optimal locations rather than the locations permitted by cabling.

1.4.3 Developing world communications

In the developing world, access to learning materials such as books and schooling is limited. Many believe that mesh networks can provide an inexpensive link to the Internet; allowing the developing world to benefit from the Internets learning resources. Furthermore, in many rural areas, a satellite link may be the only link to the rest of the world. Mesh networks allow this communications link to be shared by extending access in a hop-by-hop wireless manner. The OLPC (One Laptop Per Child) project [6] is a prime example of mesh networks providing communications to the developing world.

1.4.4 Disaster recovery and rescue services

When hurricane Katrina hit New Orleans in 2005 a large portion of the communications infrastructure was destroyed. To provide phone communications and digital maps to aid workers, a 802.11 wireless mesh network was quickly setup. Following the cleanup, the mesh network was expanded to provide communications to residents and businesses in an effort to restore the area. It is believed that mesh networks could be integral for disaster

recovery and rescue services. A 4.9 GHz public safety band has been created in many countries to allow the use of 802.11 technologies for these purposes.

1.5 Problem Definition

Despite strong fundamental reasons and a multitude of potential applications, existing data communications technologies make the implementation of multi hop ad hoc networks problematic. The OSI networking model [7] has been highly successful and many believe has played a significant role in the proliferation of an open and interoperable Internet [8]. This thesis does not advocate the abandonment of the OSI architecture, but, before multi hop ad hoc networks are feasible, some of the original design intentions of this model must be blurred. This PhD investigates the areas of routing, 802.11 multi hop contention and acknowledgment inefficiencies.

Routing is a network layer function that traditionally operates with hierarchical addressing. To enable wireless networks to self-form and self-heal, the addressing structure of these networks must be flat. The use of IP routing protocols means that hierarchical addressing, which is supposed to occur at the network layer, will be lost. The alternative is to route at the data link layer where the addressing structure is naturally flat, however, this is equally problematic because the data link layer is supposed to implement STP (Spanning Tree Protocol). Whether routing at the network layer or the data link layer, the original design goals of the data link and network layer will be blurred. Chapter 2 poses the question of the best approach to routing in multi hop ad hoc networks. The problems with traditional routing protocols such as RIP and OSPF are initially detailed. The literature is then used

to illustrate the solutions to these problems and describe the current state of the art routing protocols. An experiment is performed to answer a number of questions. These include; what routing techniques are most effective and which OSI layer is optimal?

802.11 is an ideal technology for the multi hop architecture because it is cheap, mass produced, has been the focus of extensive research efforts, and is already embedded in many phones, PDAs, DSL/cable routers, TVs, Nintendo, and fridges. Although these are strong arguments for the technology, 802.11 was never designed for a multi hop architecture because all transmissions occur over one channel, causing large contention problems. Many have suggested MAC layer modifications to make use of the multiple channels and alleviate multi hop performance problems; however, deviating from the current standards based design would mitigate many of the previously mentioned advantages of 802.11. Chapter 3 proposes a solution using multiple ad hoc radios. A channel selection mechanism is devised to intelligently assign channels to radios. Using multiple radios on different channels enables multiple concurrent transmissions, reducing media contention. This chapter develops a novel channel selection mechanism that is tightly integrated with the OLSR (Optimized Link State Routing) protocol. The solution is tested in an 8 node wireless testbed.

TCP is the transport layer protocol that facilitates fair, efficient and reliable transfer of data across the Internet; however, as it was designed for reliable wired networks, it misinterprets all losses as congestion. Subsequently, wireless links must implement their own reliability mechanisms to prevent interference or collision based losses from being detected by TCP. Unfortunately, these link layer mechanisms introduce a significant overhead;

reducing throughputs. Chapter 4 investigates alternative reliability solutions such as 802.11 BlockAck, TCP Westwood, and PEPs (Performance Enhancing Proxies) such as Snoop and Split TCP. Many of these mechanisms are either ineffective or unusable in multi hop ad hoc networks. A new PEP, known as D-Proxy, is experimentally developed and tested.

This thesis presents experimental work. There has been a large amount of conceptual and theoretical work for multi hop ad hoc networks. Seminal theoretical works such as Gupta and Kumar's *The Capacity of Wireless Networks* [9] have been integral in understanding the degradation of capacity in multi hop networks, but, the same authors subsequent experimental work demonstrates large discrepancies between conceptual estimates and actual performance [10]. Furthermore, other work also suggests that many of the simulated studies in multi hop ad hoc networks are not repeatable, and do not present realistic levels of performance [11]. Despites these criticisms, some research in mutli hop ad hoc networks has performed both simulated and experimental work and found similar results [12]. The ultimate advantage of experimental work is that the results are real world. Experimental work can also discover issues requiring attention in future work. Knowledge of these practical phenomena and issues in experimental work can be used to increase the realism of simulated work.

Increased range, throughput, power efficiency, spectral efficiency and a plethora of new potential networking scenarios have motivated extensive research in multi hop ad hoc networks. However, to provide self-forming and self-healing characteristics, the most basic network functions of multi hop ad hoc networks necessitate a distortion of the layering functions in the OSI networking model. This thesis explores the existing technologies operating

within these layers which perform poorly in multi hop ad hoc networks. It develops and evaluates solutions to these performance problems with a cross layer approach.

Chapter II

Routing Protocols in Multi Hop Ad hoc Networks

2.1 Introduction

This chapter investigates routing which is the most central, important, and well researched concept in multi hop ad hoc networks. These networks feature constantly changing link conditions, unreliable or lossy links as well as CPU and bandwidth constraints [13, 14, 5]. Combined, these features make routing in multi hop ad hoc networks a significant challenge. Our initial goal is to explain why traditional routing protocols cannot operate in wireless mesh networks. After describing the problems, possible solutions will be examined. It is hoped that describing the problems and the solutions will provide a framework for understanding new ad hoc specific routing protocols. A controlled experiment will then compare OLSR (Optimized Link State Routing) [15, 16, 17], Babel [18], and both IP based and MAC based BATMAN (Better Approach To Mobile Ad hoc Networks) [19]. This chapter will compare the different approaches. It aims to discover which OSI layer is better suited to routing. The findings will also be used to determine the most pressing challenges faced by ad hoc routing protocols.

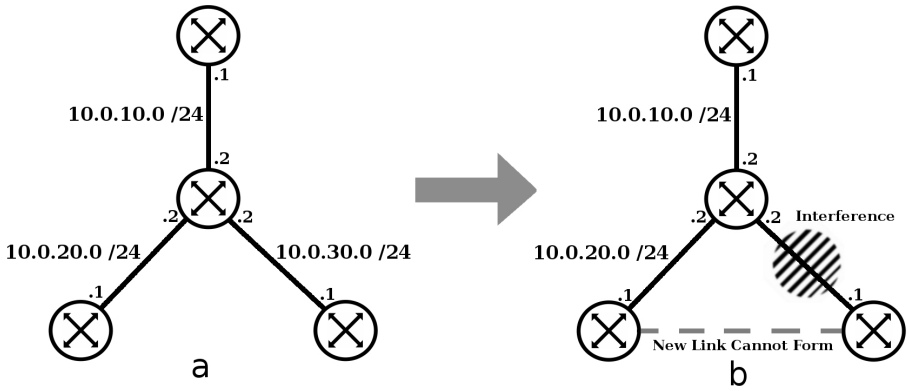


Figure 2.1: Network routes cannot adapt to physical layer changes

2.2 Challenges in Mesh Routing

2.2.1 Topology changes

Multi hop ad hoc networks are designed to have self-forming and self-healing properties to deal with topology changes. Given the failure of links or nodes, the network must automatically reform. These requirements prevent traditional hierarchical IP based addressing from being used. In traditional wired networks, directly connected interfaces are configured with IP addresses in the same subnet. The subnet mask assigned, must form a unique network address and may not be reused in the network. Hierarchical addressing schemes will not work in mesh networks.

Figure 2.1a shows three networks and a traditional assignment of IP addresses to interfaces. Upon the introduction of some interference, shown in Figure 2.1b, the lower right router will lose connectivity. However, it will be unable to form a new wireless link with another router because its interface is not configured for communication on that subnet. The IP configuration, shown in Figure 2.1b, prohibits the formation of a new link.

The solution to this problem is to use a flat addressing structure. As a result, instead of a traditional routes such as:

10.0.20.0/24 via 10.0.10.2

Mesh routing protocols must route using host addresses.

10.0.0.23/32 via 10.0.0.3/32

Equally, MAC addresses could also be used:

00:16:05:25:B2:F0 via 00:48:DE:45:82:C4

There is considerable debate as to whether routing in mesh networks is better performed at the network layer or data link layer with the IETF [4] and IEEE [20] both working on independent routing protocols. These discussions will be saved for section 2.6.1.

Regardless of the layer, flat addressing enables adaptable self-forming and self-healing properties, however, many of the advantages implicit in hierarchical addressing are lost. For example, no mechanism exists to prevent broadcasts which may now flood throughout the network. Address summarization is another feature that is unable to be used by mesh routing protocols. Routing with a flat addressing scheme will require more routing entries

compared with traditional hierarchical addressing because a routing entry will be required for every node rather than a subnet of hosts. Furthermore, given the changing conditions surrounding mesh nodes, these networks may require frequent updates and impose heavier CPU loads from routing table look-ups.

2.2.2 CPU and bandwidth constrained environments

Network devices in wireless mesh networks are likely to be both CPU and bandwidth constrained [5]. The wireless nodes performing route calculations are often low power, low cost embedded machines. The CPU power of such nodes will therefore be restricted. In addition to being CPU constrained, these devices will also operate using WiFi chips which offer less bandwidth than equivalent wired links. These bandwidth limitations will be exacerbated in dense networks where the actual throughput is only a fraction of the data rate due to media contention.

2.2.3 Constantly changing link conditions

It is commonly asserted that traditional routing protocols such as RIP and OSPF update too infrequently to deal with the constant changes that occur in multi hop ad hoc networks. Because traditional protocols update infrequently, they can only accept a few topology changes per minute [21]. The constantly changing conditions in mesh networks necessitate a frequent

Protocol	Reliable Updates	Hello Int	Top Exch Int	Triggered Updates
RIP	No	-	30 secs	Yes
EIGRP	Yes	5 secs	Infinity	Yes
Babel	No	4 secs	20 secs	Yes
OSPF	Yes	10 secs	30 minutes	Yes
OLSR	No	2 secs	5 secs	No

Table 2.1: Hello and topology exchange intervals of routing protocols

stream of hellos and topology exchanges to track the changing link conditions. Table 2.1 compares some notable attributes of traditional and mesh routing protocols. Note that the two ad hoc routing protocols, Babel and OLSR (Optimized Link-State Routing), have significantly lower hello and topology exchange intervals than the current wired routing protocols. Overheads will therefore be higher in ad hoc rather than traditional routing protocols [18].

2.2.4 Unreliable links

A secondary reason why updates must occur more frequently is because wireless networks are less reliable than traditional wired networks. In wireless mesh networks, topology changes or hellos may frequently be dropped. Given that wireless networks are unreliable, it is counter intuitive that the mesh routing protocols such as OLSR and Babel use unacknowledged updates (Table 2.1). The reason that unacknowledged updates are used is due to the cost of reliable updates. Reliability in a wireless environment would require individual unicast acknowledgments to be sent for every update received. In a shared media broadcast based wireless environment, large numbers of

unicast acknowledgments would dramatically increase the overhead of the routing protocol [18]; particularly in dense environments. Given that multi hop ad hoc networks constantly change, this would introduce an enormous overhead.

In link-state routing, the Dijkstra algorithm provides 100% loop freedom as long as the link state databases are synchronized. Thus, the reason reliable routing information is critical is because desynchronization, which can be caused by lost updates, results in routing loops. Unicast acknowledgements to updates are infeasible because of the huge overhead they would impose in a shared medium environment.

A consequence of this unreliability is that mesh routing protocols must rely on frequent and redundant routing table updates rather than reliable updates. Subsequently, they generate more traffic than traditional protocols [18] such as such as OSPF (Open Shortest Path First) [22] or EIGRP (Enhanced Interior Gateway Routing Protocol) [23]. A possible solution being discussed by the developers of OLSR [24] is to implement database consistency checks rather than reliable topology dissemination.

2.2.5 Count to infinity problem

Distributed Bellman-Ford is the original algorithm used in RIP (Routing Information Protocol) for computing routes to destinations. This protocol relies on neighbors informing one another of their distance to different destinations. In its original state, the bellman-ford algorithm can cause count-to-infinity routing loops. The well known RIP routing protocol is an implementation of the distributed Bellman-Ford algorithm which uses split

horizon, hold down timers, route poisoning and triggered updates to prevent count-to-infinity loops. Under specific situations, routing loops can still form. This led to the development of the DUAL algorithm [25], popularized by Cisco's proprietary routing protocol; EIGRP (Enhanced Interior Gateway Routing Protocol) [23]. Despite these improvements, neither RIP or EIGRP are usable in wireless mesh networks.

One reason why distance-vector routing protocols such as RIP and EIGRP are poor candidates for mesh routing protocols is due to the way they deal with the count-to-infinity problem. Before describing the inadequacy of RIP or EIGRP's solution, the count to infinity problem will be explained.

Two routers are exchanging periodic updates in Figure 2.2a. In Figure 2.2b, router B's locally connected subnet 10.2.2.0 goes down. Before router B is able to inform its neighbors, Router A re-advertises the 10.2.2.0 route back to B with a metric of two. Node B then wrongly believes that A must have a alternate route to 10.2.2.0 with a metric of two. Node B then adds a route to 10.2.2.0 via router A with a metric of three (Figure 2.2c). When A no longer receives any advertisements to 10.2.2.0 with a metric of one. It will age this route out of its routing table. However, router B will then advertise that it contains a route to 10.2.2.0 with a metric of three and router A will re add the route to 10.2.2.0 with a metric of four (Figure 2.2d). This process will continue until infinity, or, in the case of RIP, until the maximum hop count of sixteen (Figure 2.2e). This problem is commonly known as the count to infinity problem because this process will continue to infinity or the maximum hop count.

RIP and EIGRP employ numerous features to prevent count to infinity routing loops. One feature is triggered updates, which will cause route

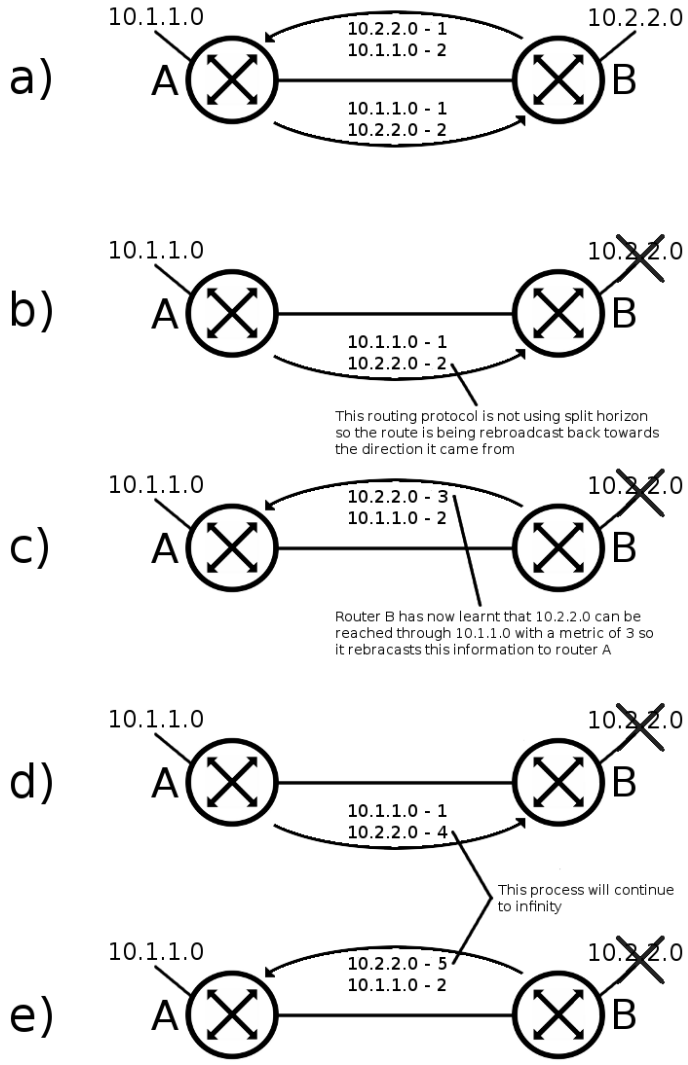


Figure 2.2: Count to infinity problem

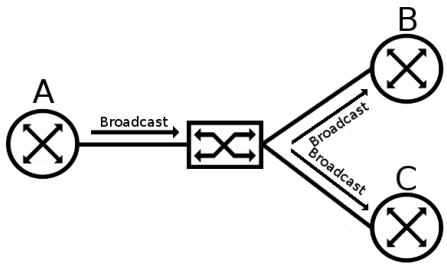


Figure 2.3: Split horizon in wired networks

changes to instigate/trigger an immediate route update. Another feature is hold down timers which are used to ensure that after a local route is removed from the routing table, any updates for the route will be ignored until the expiry of the hold down timer. The final feature is called split horizon.

The split horizon rule states that a route should never be re-advertised through the same interface that the route was received. In the wired network shown in Figure 2.3, the split horizon rule is sensible because stations that are attached to the same Ethernet segment will hear every broadcast. However, in wireless mesh networks, with multi hop topologies, wireless nodes *must* rebroadcast routing information over the same interface. In Figure 2.4, node B must rebroadcast the update from Node A to ensure Node C is informed of the route. This example shows that split horizon may not be used in wireless mesh networks. If split horizon was employed, node C could never learn of node A. To summarize, traditional distance vector routing protocols are inappropriate because they prevent count-to-infinity routing loops by using the split horizon rule.

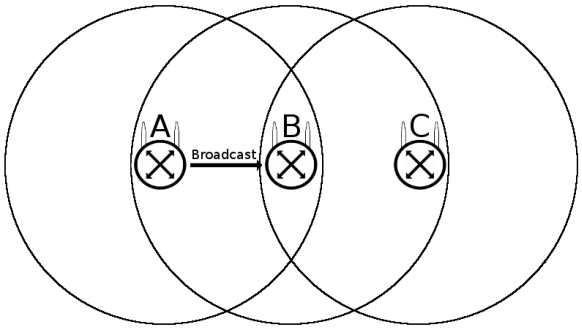


Figure 2.4: Split horizon in multi hop ad hoc networks

2.2.6 Incompatible link state interface types

The traditional link state protocols such as Open Shortest Path First (OSPF) [22] are equally inappropriate for wireless mesh networks. The network types that have been designed for existing wired networks, namely; point-to-point, broadcast, non-broadcast multi access, point-to-multi-point and virtual, do not meet the requirements of wireless mesh networks. Wireless mesh networks are a broadcast based technology, but, the broadcast network type provided in OSPF is inappropriate. Figure 2.5 shows that the OSPF broadcast interface type elects a Designated Router (DR) and Back-up Designated Router (BDR). All OSPF routers within this Ethernet network maintain adjacencies with these two routers. The purpose of this mode is to reduce OSPF's overhead by reducing the number of adjacencies. The problem with using the broadcast network type is that all nodes must be able to *directly* contact each other, normally known as a full mesh. As demonstrated in Figure 2.3 and Figure 2.4 these concepts taken from wired broadcast networks do not work because wireless mesh networks are fundamentally multi

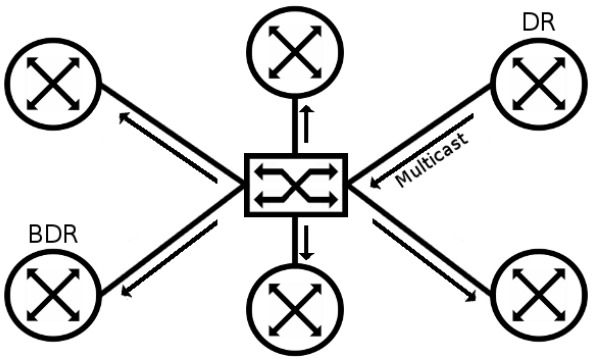


Figure 2.5: OSPF using the broadcast interface type

hop environments.

The OSPF point-to-point network type is also inappropriate because it requires one interface for every neighbor. To use this network type, wireless nodes would require one 802.11 interface for every neighbor. The point-to-multi point interface circumvents this limitation, allowing each interface to have many neighbors, but shifts the problem to scalability. The point-to-multi-point network type uses a series of point-to-point interfaces to maintain adjacencies with every neighboring node. Henderson et al [26] found that using this interface type in mesh networks led to prohibitive amounts of overhead with as few as 20 nodes. These problems have stimulated interest in the development of a new interface type for OSPF known as a MDR (MANET Designated Router) [27]. The convenience of operating a mesh routing protocol without redistributing external routes from the wired network is the principle reason for the interest [26, 28, 27, 29].

2.2.7 Hierarchical/scalable routing techniques

The traditional hierarchical routing techniques that are used to provide

scalability also become problematic. OSPF has a form of hierarchical routing based on areas and Intermediate System to Intermediate System (IS-IS) uses multiple levels to create hierarchies. These approaches have successfully allowed OSPF and IS-IS networks to scale efficiently. Using any of these hierarchical schemes is impossible in wireless mesh networks because they are designed for static networks and require administrator configuration. These features are incompatible with the self-forming and self-healing requirements of wireless mesh networks.

2.2.8 Metrics

Routing metrics are crucially important to the performance of any routing protocol. One of the commonly stated reasons why OSPF is a better routing protocol than RIP is because its metric can incorporate bandwidth rather than just hop count. Like many other mechanisms in OSPF, the bandwidth of a link is an administrator configured variable. Similar to the hierarchical routing mechanisms, metrics that require administrator configuration are inappropriate. Creating efficient metrics which can dynamically adapt to the conditions in wireless mesh networks is both conceptually and practically difficult.

2.2.9 Summary of the mesh routing challenge

To reiterate the challenges, mesh routing protocols must find a solution to the count to infinity problem, or, if using link state routing, must develop an entirely new interface type for wireless mesh networks. In addition to these

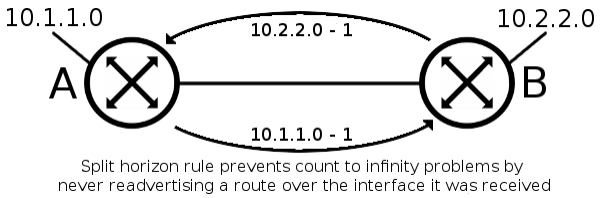


Figure 2.6: How split horizon prevents count to infinity problems

problems, new ad hoc routing protocols must be able to accept a greater number of changes over less reliable links. These routers must also operate with less CPU power and over low bandwidth links. A flat addressing structure must be used which means that a route will be required for every individual node. As every node will require an individual route, it will be likely that routing tables may be large and subject to frequent changes. Traditional, area or level based approaches, administrator configured metrics and address summarization are inappropriate due to the self-forming self-healing requirements of mesh networks [30]. This makes the goal of routing in multi hop ad hoc networks highly challenging.

2.3 Solutions in Mesh Routing

2.3.1 Solving the count to infinity problem

As previously stated, the count to infinity problem cannot be solved with the split horizon rule. The split horizon rule, shown in Figure 2.6, prevents count to infinity loops by stopping routing updates, received on one interface, from being rebroadcast back over the same interface. Unfortunately, in wireless mesh networks, nodes require this function to relay routing messages.

In 1994, Perkins et al [13] proposed that routing updates be appended

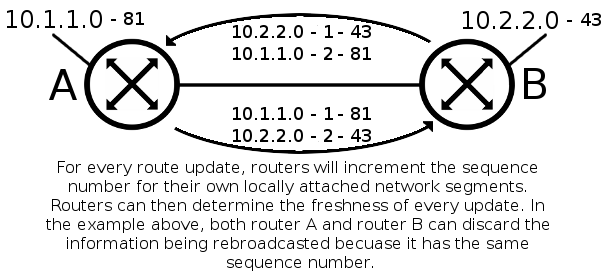


Figure 2.7: How sequence numbers prevent count to infinity problems

with sequence numbers. When sequence numbers are used, nodes rehearing the original re-broadcasted route can identify the freshness of the update. A router will only prefer a given route if the sequence number of the route advertisement is newer, indicating a fresher route, or, if the route being re-broadcasted has the same sequence number but a better metric. Evident in Figure 2.7, a node that is receiving a route that was originally sent through itself can never have a better metric. Perkin's seminal paper [13] introduced this idea alongside the DSDV (Destination Sequence Distance-Vector) routing protocol; which is an adaptation of RIP for ad hoc networks. In the subsequent years, newer ad hoc routing protocols such as DSR (Dynamic Source Routing) [31] and AODV (Ad-hoc On demand Distance Vector) [32] reused this idea. However, these newer routing protocols began to focus on a new approach known as reactive routing.

2.3.2 Reactive routing

The commonly understood traditional routing protocols such as RIP, EIGRP, OSPF and the previously mentioned DSDV are proactive routing protocols because they proactively build and maintain routes to all destina-

tions. In ad hoc networks, it is arguable whether the continual maintenance of routes to destinations, some which may never be used, is wasteful of CPU cycles, bandwidth and energy [14]. This idea lead to the exploration of a different, reactive routing protocol. The reactive idea is that it is better to search for routes when they are required, rather than undergo the constant overhead of maintaining routes to destinations which may never be used. This fundamental idea has both advantages and disadvantages.

Reactive routing protocols, incur little overhead when the network is idle [33, 14]. In the presence of mobility, reactive routing will also have lower overheads than proactive routing [30]. The reactive routing protocols were originally believed to be ideal for mobile ad hoc networks and devices with less CPU, memory and batteries than dedicated network devices [14]. The major disadvantage of reactive routing protocols is the initial look-up latency. As routers do not have routes stored in routing tables, routes must be discovered before they can be used.

Reactive routing protocols learn routes when required by flooding route requests; hereon referred to as RREQs (Route REQuest). These RREQs are responded to by either the destination node or by a node with a current route to the destination. The RREQ is responded to by a RREP (Route REPly). The RREP informs the RREQ initiator of the best route to the destination. Following the return of the RREP, data packets may be forwarded. In the event of a link break, a RERR (Route ERRor) packet is transmitted to all nodes participating in the route. This will cause the route to be removed from the routing tables and, if there is still data waiting to be sent, initiate another RREQ in search of another route to the destination. The use of RREQ's and RREP's is diagrammatically shown in Figure 2.8.

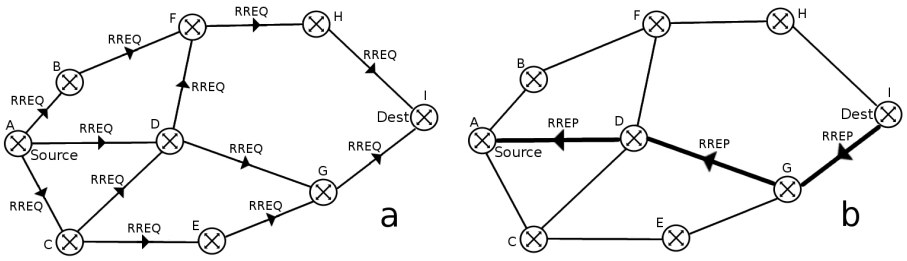


Figure 2.8: RREQ (Route Request) and RREP (Route Reply)

2.3.2.1 Source routing vs hop-by-hop routing

A debatable architectural consideration is whether reactive routing should be source routed or routed hop-by-hop. The familiar mainstream routing protocols such as RIP and OSPF are hop-by-hop routing protocols. In hop-by-hop routing, each router receiving a packet independently evaluates the lowest cost path and forwards the packet to the next router. Routing decisions are therefore made on a hop-by-hop basis and each router is only responsible for passing the packet to the next router. In source routing, the sending node, or source, is responsible for specifying the entire route in advance. Each data packet being source routed must therefore have the address of all intermediary nodes between the source and the destination in the header of the packet. In source routing, the intermediary routers receiving packets are not passing the packet based on their own routing information, but instead based on the routing information specified in the header of the packet. Subsequently, source routing protocols do not have routing tables; but route caches. These route caches are only used by the node sourcing the packet.

Source routing protocols require every packet to contain a routing spe-

cific header. The size of this header will increase with the number of nodes between the source and destination. Perhaps the biggest advantage of using source routing is that loops are very easy to detect and remove. Routes which feature the same router twice in the header are obvious loops. Source routing requires less CPU power to run as routers are not constantly performing routing table look-ups [33]. Another large benefit of source routing is path accumulation [33].

2.3.2.2 Path accumulation

Figure 2.9 and Figure 2.10 show the process by which source routed DSR and hop-by-hop routed AODV learn routes. Recall that in source routing, every packet contains the entire route in the header of the packet. This means that DSR or source routed nodes are able to learn a large amount of network information. The path accumulation feature present in DSR is compared with hop-by-hop AODV routing in Figure 2.9 and Figure 2.10. These figures illustrate how DSR nodes are able to use path information to learn routes to every other node on the path. Comparatively, AODV nodes, which do not use path accumulation, only learn of the next hop on the path. The greater network knowledge provided by path accumulation in DSR can increase the speed of the RREQ/RREP process and reduce overheads.

Greater network knowledge can reduce overheads because *any* node with a route to the destination can respond to a RREQ. For example, in the Figure 2.11, node B, which has already learnt of the route to node D, is responding to node A's query for a route to D. Greater network knowledge can also reduce the initiation of RREQs and RREPs because in many cases, nodes

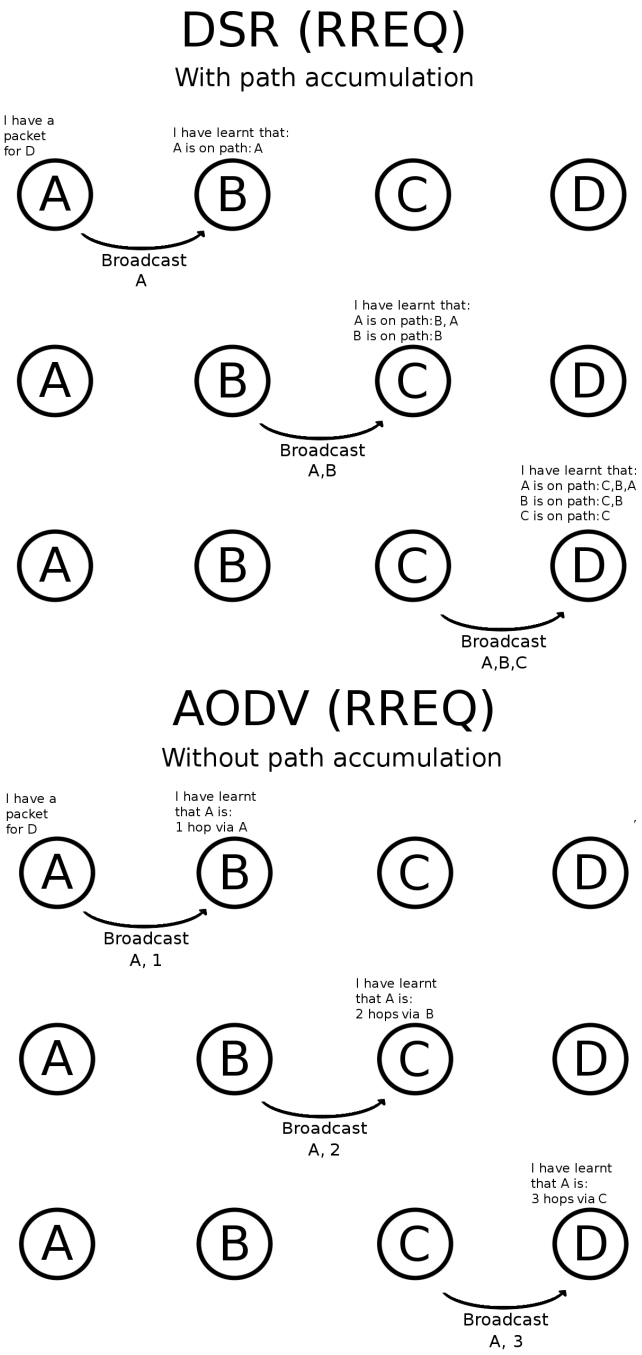


Figure 2.9: Path accumulation in DSR RREQ as compared to AODV

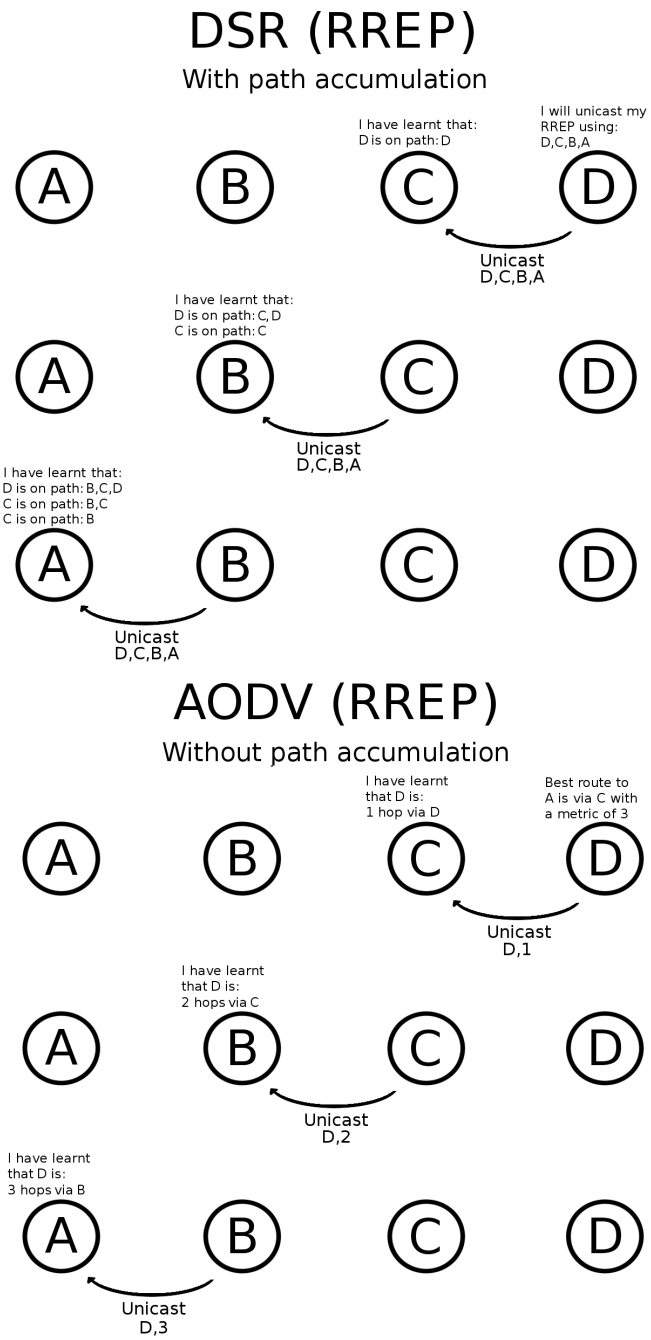


Figure 2.10: Path accumulation in DSR RREP as compared to AODV

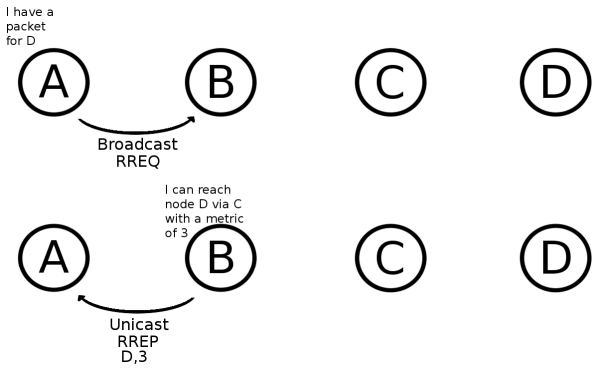


Figure 2.11: Suppression of RREQs through greater network knowledge

may have already learnt of routes to their destinations.

The negative affect is that using path accumulation will increase the size of RREQs and RREPs, however, in 802.11, the additional overhead is generally trivial in comparison to preambles, trailers and DCF (Distributed Coordination Function) [34]. The consensus was that the reduction in RREQs and RREPs is more beneficial than the minor overhead imposed. As a result, path accumulation has been incorporated into DYMO (DYnamic Manet On demand) [35]. DYMO is a new version of AODV which incorporates some functionality from DSR. DYMO is the current mainstream reactive routing protocol pursued by the IETF.

2.3.2.3 Hybrid routing

The hybrid routing protocols use a combination of traditional proactive routing and the recently described reactive routing. In ZRP (Zone Routing Protocol) [36], routes to nearby nodes are proactively maintained whereas routes to more distant nodes are discovered reactively. In Figure 2.12 node A will proactively maintain routes to all nodes in the proactive area while

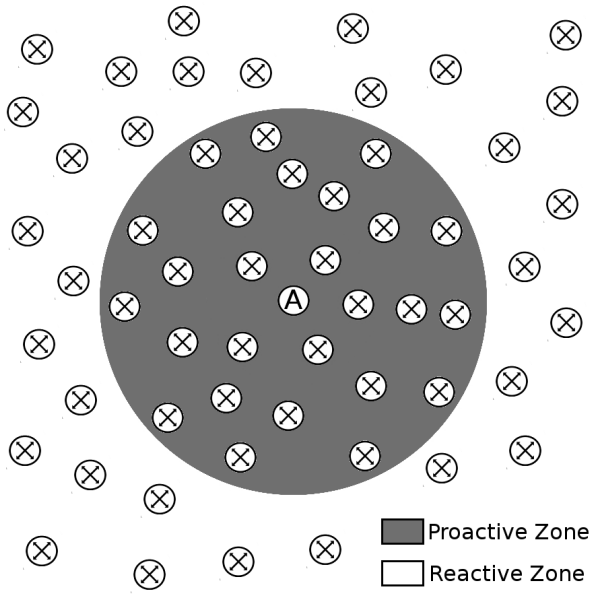


Figure 2.12: Zone Routing Protocol

nodes outside the proactive areas, will be discovered reactively. The size of the proactive zone is designed to be dynamically adjusted from fully proactive to fully reactive based on network conditions. No working implementation of ZRP currently exists and thus it may currently be too complex for actual implementations. The first actively pursued hybrid routing implementation is HWMP (Hybrid Wireless Mesh Protocol) which is a protocol devised by the IEEE TGs [37].

2.3.3 Limited dissemination

A popular approach to reduce routing overheads in both proactive and reactive protocols is to limit the dissemination of routing information. The origins of limited dissemination techniques were founded in Distance Routing Effect Algorithm for Mobility (DREAM) [38] which is a position based rout-

ing[1] protocol. The observation described in DREAM is that distant nodes appear to move more slowly than nearby nodes. DREAM reduces network overheads by updating distant nodes less frequently than nearby nodes.

The application of this concept to link state routing is commonly known as Fish eye State Routing (FSR) [39]. These techniques have been shown to significantly reduce overheads [40]. The reason that imprecise or slightly inaccurate information can be tolerated is because routing decisions are made on a hop-by-hop basis. This means that if a node is many hops away, a route in the general direction will often suffice. This imprecise knowledge is compensated by the fact that as a transmitted packet gets closer to the destination, routing decisions become more accurate.

Fish eye state routing modifies the TTL (Time to Live) in routing messages to update nearby and distant nodes at different intervals. This concept is illustrated in Figure 2.13.

Studies have shown that FSR provides more optimization in networks with a large diameter [40]. While FSR can reduce the number of link state messages, it can lead to suboptimal routes. The trade-off between suboptimal routes and lower overheads requires consideration [41]. This idea is explored by Santivanez et al [42] who searches for the optimal update frequency and dissemination distance. The inclusion of this technique into OLSR implementations [24] is a testament to its effectiveness.

The application of limited dissemination techniques in reactive routing protocols such as AODV [43, 44], and DYMO [35], are known as the expand-

[1] Position based routing is not considered for 802.11 based wireless mesh networks because they are designed for long range communication technologies. With 802.11 position does not necessarily indicate a good route because of the impact of physical obstructions on small hops.

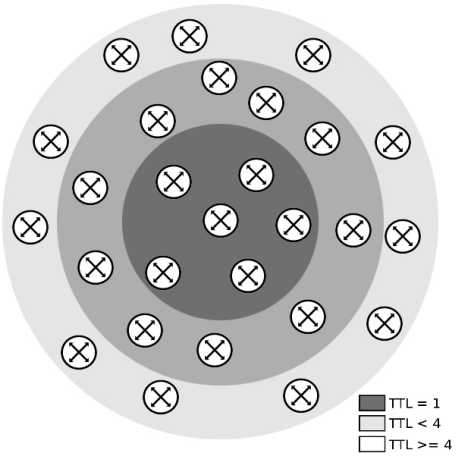


Figure 2.13: FSR (Fish-eye State Routing)

ing ring search [43]. The implementation is similar to FSR, using increasingly higher TTLs in the RREQs. The idea was that a single RREQ, which was usually broadcasted though the entire network, should be initially limited to a number of hops. When broadcasting RREQs, the destination may be only a few hops away or a nearby node may posses a usable route to the destination. In this case, a nearby node can reply to the RREQ and the expanding ring search technique will prevent the broadcast from traversing the entire network. Limited dissemination techniques lower overheads and improve the scalability of the routing protocol [41]. They can also be combined with efficient dissemination techniques to further reduce update overheads in proactive routing protocols.

2.3.4 Efficient dissemination

Unoptimized link state algorithms produce prohibitive amounts of overhead in ad hoc networks with only 20 nodes [26]. Due to the large number

of adjacencies formed in ad hoc networks, there are a large number of redundant transmissions during flooding. The broadcast based DR/BDR system designed for wired OSPF broadcast networks cannot translate to wireless multi hop networks because it assumes that all nodes can *directly* communicate with one another.

The replacement system for multi hop ad hoc networks involves electing or selecting relay nodes that are responsible for flooding link state messages. The technique is better explained diagrammatically. Given the routers shown in Figure 2.14, an efficient dissemination technique will try to find a set of nodes that can efficiently disseminate topology information to surrounding nodes. In Figure 2.15, the Grey routers have been elected as Multi Point Relay (MPR) nodes to broadcast topology information to the rest of the network. Finding the minimum set that can be chosen as relay nodes is more efficient, however, it is also a NP hard problem [45]. In the OLSR protocol, this system is called MPR. By restricting the number nodes responsible for flooding, the number of redundant transmissions is minimized [46]. Individual MPRs are elected by surrounding nodes. The details of this process can be found in [16] and [17].

Figure 2.16a shows a link state flood without MPRs. In classical flooding, every node individually rebroadcasts topology information; causing an enormous number of redundant transmissions. In Figure 2.16b, the Grey routers have been elected as MPRs. These MPRs are solely responsible for rebroadcasting routing information. Figure 2.16 illustrates how efficient dissemination techniques can dramatically reduce the number of redundant transmissions. MPRs increase the scalability of link state algorithms, especially in dense environments [40], and are an essential part of the OLSR [16]

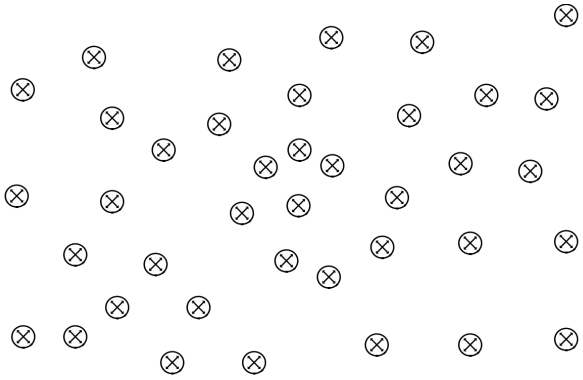


Figure 2.14: Selection of MPRs I

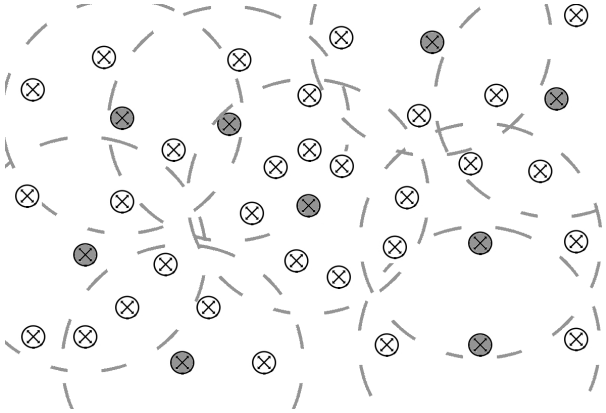


Figure 2.15: Selection of MPRs II

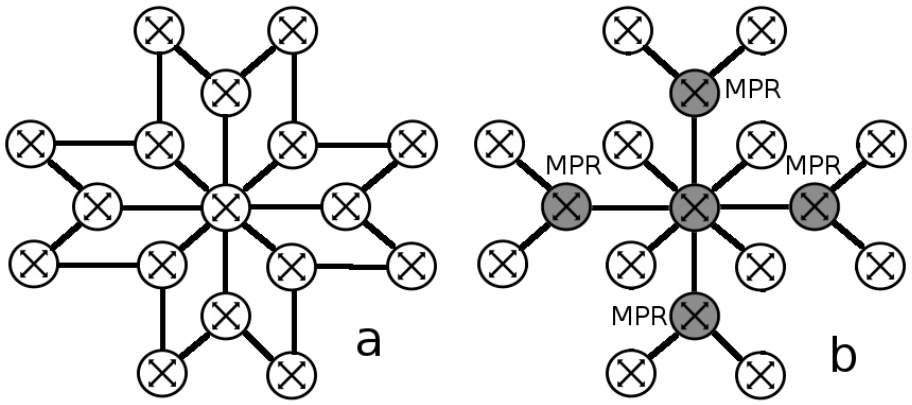


Figure 2.16: Efficiency of MPRs (Multi Point Relays)

routing protocol.

The currently proposed extension to OSPF [27] introduces a new network type called MANET Designated Router (MDR). It is similarly based on efficient dissemination of topology information by electing relay nodes. MDR [27] uses a different selection algorithm based on Connected Dominating Sets (CDS). There are claims that the minimal CDS algorithm is superior to MPR, however this is subject to debate [47].

2.3.5 Routing metrics

Path selection in routing protocols is based on routing metrics. In wireless mesh networks, the self forming, self healing characteristics mean that variables such as bandwidth and delay cannot be manually entered as they are in OSPF or EIGRP. As a result, designing routing metrics for multi hop ad hoc networks is a difficult endeavor.

2.3.5.1 Hop count

Hop count simply counts the number of links between the source and the destination and chooses the path with the least number of hops. The hop count metric is used for simplicity, not performance. Many routing protocols such as DSR [31] and AODV [32] use hop count as a metric. The limitations of using hop count as a metric are well known. The traditional problem is shown in Figure 2.17. In this network, data transmissions between node A and node C will traverse the 256 Kb/s link rather than the two 100Mb/s links.

These problems worsen in wireless mesh networks because paths with fewer hops are likely to be routes between distant and therefore lower data rate links. In many cases this will lead to routers preferring longer distance, lower speed links over shorter, higher throughput links. This is shown in Figure 2.18 where the long distance 6 Mbit link will be the preferred path between A and C. Kawadia et al [8] state that this unintentional cross layer interaction leads to performance degradations. Couto et al [48] found that hop count performed poorly in static mesh networks. In contrast to these results, Draves et al [49] found that, in highly mobile environments, hop count outperforms other metrics such as Expected Transmission Count (ETX); discussed in section 2.3.5.4.

2.3.5.2 Per-hop RTT (Round Trip Time)

Per-hop RTT [49] is a round trip time based metric. Nodes will transmit probes and measure the time taken for a reply. A exponentially weighted

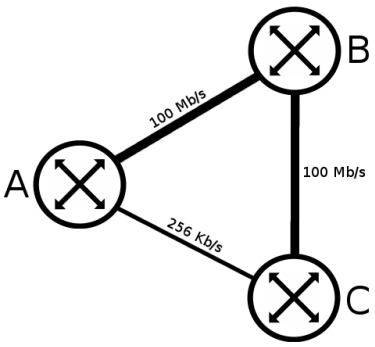


Figure 2.17: Traditional hop count problem

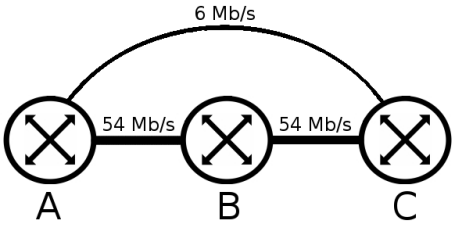


Figure 2.18: Wireless hop count problem

moving average is created from the measured variable. Per-hop RTT captures numerous network variables including queuing delay, media contention and loss. Queuing delay is captured by the RTT metric because the RTTs will be higher if the request or reply is queued in a buffer awaiting transmission. Media access can also affect the metric because large numbers of stations waiting for media access will increase the Per-hop RTT. Finally, loss is also measured because, if a packet is corrupted and must be retransmitted by the link layer, it will incur additional delay. If losses are so bad that link layer retransmission mechanisms are unable to retransmit the lost packet, the average Per-hop RTT is increased by 20%. Upon initial inspection Per-hop RTT is an ideal metric because it is able to capture a variety of network attributes that affect performance; however, it is load dependent.

In 1989, Atul Khanna [50] published a paper discussing how load depen-

dent metrics cause route oscillations. The conclusion of a more recent study [49] of metrics, specifically in mesh networks, found similar results. Draves et al [49] found that self interference, was the cause of route instability and showed that Per-hop RTT is outperformed by both ETX and hop count. Self interference occurs because the sender side queuing delay is an inordinately large contributor to the RTT. When a router chooses a particular route and starts moving data packets over that link, it will (self) interfere with that metric. Put simply, links chosen by Per-hop RTT will experience increases in queuing delays and therefore increases in the the Per-hop RTT. This self interfering property causes route oscillations.

2.3.5.3 Per-hop Packet Pair Delay (PktPair)

Per-hop packet pair delay [49] is conceptually similar to Per-hop RTT. In PktPair, two packets are sent to a neighboring node; a small initial packet, followed by a second large packet. PktPair measures the delay between the first small packet and the second large packet and adds it to a exponentially weighted moving average. Similar to Per-hop RTT, PktPair is able to capture loss and media contention. In addition, it can also measure bandwidth because higher bandwidth links will serialize the second larger packet faster; resulting in lower and more favorable metrics.

The previous metric, Per-hop RTT [49], had a serious flaw with self interference because once a link was selected as the best route, queuing delays and hence RTTs increased. Unused links would experience very small queuing delays. With no packets being routed over the interface, queuing delays are minimal; resulting in low RTTs. This relationship causes route oscilla-

tions. PktPair can avoid the affects of queuing delays which were the cause of oscillations in Per-hop RTT because the packet pair is sent back to back. The variable measured is not the RTT but the time difference between the return of the first and second packet. PktPair should therefore be immune from queuing delays and thus, self interference, because the packet pair enter the queue back to back.

Unfortunately, in the context of multi hop ad hoc networks, this technique is not immune from self interference [49]. The source of self interference in PktPair is caused by the MAC (Media Access Control) function. 802.11 is a half duplex technology and is unable to simultaneously send and receive. Following the transmission of the first probe, one of two outcomes are possible. The second packet PktPair probe will get priority and transmit, or, the destination will get priority and return the first initial probe to the sender. Regardless of the outcome, the inability of 802.11 to simultaneously send and receive will skew the results.

Another unrelated problem is the overhead of sending two packets to measure links statistics. Mindful that one of these packets is a large packet, such a metric would introduce additional overheads into the network. These problems with Per-hop RTT and Pkt Pair are non-trivial. Studies have shown that they are outperformed by ETX (Expected Transmission Count) [51, 49].

2.3.5.4 ETX

ETX (Expected Transmission Count) [48] is a reliability metric designed to overcome some of the limitations of the hop count metric. The ETX metric finds paths requiring the fewest transmissions. Although packets in

802.11 networks are acknowledged at layer 2; retransmissions result in a loss of airtime and hence, bandwidth. Therefore, low loss paths are a valid measurable attribute.

ETX calculates the number of successful transmissions in both directions on a wireless link. To determine these statistics, every node periodically broadcasts a configured number of probes. Receivers calculate the number of probes received against the number expected. For example receiving 8 of the 10 probes would indicate a receive probability or d_r of 0.8. However, as links are not symmetric, it is equally important to measure the success rate of their own transmissions. To obtain this information, each node will place its d_f in the probes being sent. This will inform neighboring nodes of the rate at which other nodes are receiving their probes. The formula for calculating the ETX of a link is show in 2.1. The total cost of a path will be the summation of all the links between between the source and destination.

$$ETX_l = \frac{1}{d_f \times d_r} \tag{2.1}$$

ETX messages are broadcasts which means that individual probes do not have to be sent individually to all neighbors. Furthermore, as 802.11 does not use link layer reliability for broadcasts, lost ETX messages will not be recovered. This ensures that the ETX metric obtained should be indicative of the number of transmissions required to move a packet from the source to the destination. Unfortunately, this statement is not entirely true.

ETX probes, which are sent as broadcast packets, use the baseline data rate. This data rate is usually more robust than the rate being used for unicasts. The unicast loss rate may therefore be higher than indicated by

ETX. Another problem with ETX is that wireless networks often suffer from burst packet losses rather than evenly distributed packet losses [52]. Subsequently, averaging packet losses over a small time period may give an overly optimistic or pessimistic view of losses at different time periods.

Perhaps the biggest problem is that ETX does not consider the bandwidth of the link and may favor a slower long distance link over a larger number of high speed links. The example shown in Figure 2.18 where a wireless device prefers a long distance 6Mb/s link over two 54Mb/s links remains an area of concern for ETX. Despite numerous problems, ETX is used by OLSR [17] and Babel [18]. The metrics that conceptually succeed ETX are difficult to practically implement.

2.3.5.5 ETT

The ETT (Expected Transmission Time) metric, proposed by Draves et al [53], improves on the ETX metric by adding the ability to measure bandwidth. The formula for the metric is shown in 2.2 where S is the size of the packet and B is the bandwidth. Implementations of ETT are limited because, they require inter-operation with the wireless driver. To implement ETT there must be a standardized way to obtain the data rate from the wireless driver. Until, such mechanisms are widespread, ETT implementations will be problematic and suffer from interoperability problems. ETT is a significant improvement over ETX and balances path length, transmission reliability and bandwidth. While ETT offers significant benefits over ETX, for the reasons detailed in section 2.3.5.7, it is difficult to practically implement in current routing protocols.

$$ETT_l = ETX_l \times \frac{S}{B} \tag{2.2}$$

2.3.5.6 Airtime Link Metric

The default metric in the 802.11s specification [20] is the Airtime Link Metric. This metric calculates the airtime required to transmit a 8224 bit packet between the source and the destination. The metric is formally shown in equation 2.3. The Airtime Link Metric is a more granular version of ETT. Both of these metrics measure the bandwidth of the link and the error rate. As the Airtime Link Metric was proposed by the 802.11s working group [20], it is designed to operate at the data link layer. Many of the variables such as bit rate protocol overheads are more accessible because the routing protocol is integrated with the wireless driver. In comparison, the ETT metric was proposed by a group working at the network layer [53] and thus, incorporating bit rate into the metric is problematic.

$$Ca = (Oca + Op + \frac{Bt}{R}) \times \frac{1}{1 - Ept} \tag{2.3}$$

Where:

Oca =Channel Access Overhead (PHY Dependent see Table 2.2)

Op =Protocol Overhead (PHY Dependent see Table 2.2)

Bt =Number of bits in test frame (default is a 8224 bits)

R =The bit rate of the link (e.g. 2, 24, 54Mb/s)

Ept =The frame error rate (a percentage based on the test frame)[2]

Table 2.2: Airtime variables for 802.11a and 802.11b

Variables	802.11a	802.11b
Oca	75μs	335μs
Op	110μs	364μs
Bt	8224 bits	8224 bits

2.3.5.7 Routing metric challenges

The majority of implemented routing protocols use the ETX metric. ETX's popularity is due to the difficultly of practically incorporating bandwidth into the routing metric. The architectural problem is that the bandwidth of a link is determined by the physical layer and thus this information is only available to the driver. For routing protocols to obtain these statistics, wireless drivers must be open and written to allow access in a standardized manner. Despite the proposal of increasingly complex routing protocols, the practical implementation of simpler routing metrics such as ETT is currently the biggest hindrance on performance.

2.4 Routing Protocols

This section discusses the well known routing protocols. It draws on the previously explained concepts to aid exposition.

[2] The frame error rate is implementation specific and is optional

2.4.1 DSR

Dynamic Source Routing (DSR) [31] is one of the pioneering ad hoc routing protocols and most pure forms of on-demand routing. David B. Johnson formulated the fundamental ideas in 1994 [14] but the DSR protocol details were not formally expressed until two years later [54]. The protocol has also been released as an experimental IETF RFC 4728 [31]. Testing reveals that it has lower overhead than other on-demand routing protocols such as AODV [33]. DSR's overhead is lower because there are zero periodic messages and routes are discovered on demand. This makes DSR ideal for devices powered by batteries and with low expected traffic loads.

DSR does not need sequence numbers to avoid count to infinity problems because loops are easily detected. In DSR, loops are identified when the same router appears multiple times in the route; specified in the header. Also, as previously discussed, DSR RREQs and RREPs contain a large amount of routing information. Route accumulation enables intermediate nodes to cache and utilize this information building network knowledge. DSR can cache multiple routes to a particular destination, and thus with the failure of one path, another can become instantly available.

However, there are multiple problems with DSR. As DSR has no mechanism to age out, or determine the freshness of routes, this can lead to the use of old or suboptimal routes. Instead DSR relies on the MAC layer for link failure information. Also, as the size of the network increases, the size of the routing header, which is inserted in every data packet will increase. This means that the size of the DSR header will grow with the network, increasing the size of all packets. Perhaps one of the biggest problems is the metric, hop

count, which can result in suboptimal routes. Many of the advantages and disadvantages of DSR are a consequence of using source routing. AODV was an attempt to couple the reactive routing concept in DSR with hop-by-hop routing.

2.4.2 AODV/DYMO

Ad hoc On demand Distance Vector (AODV) followed DSR and was first proposed in 1997 [55], however, is was not finalized as an experimental IETF RFC until years later [32]. AODV is an on demand routing protocol that utilizes features from both DSR and DSDV. The On-demand RREP and RREQ mechanism is adapted from DSR while the hop-by-hop distance vector routing and the calculation of routes are borrowed from DSDV. The use of sequence numbers to avoid count to infinity loops are also borrowed from DSDV.

Unlike DSR, which uses source routing, the use of hop-by-hop routing allows AODV to operate without any routing specific headers. AODV maintains connectivity with its directly connected neighbors using periodically broadcasted hellos. These messages are RREQs with a Time To Live (TTL) of 1 and are used to detect breaks in connectivity.

As a result of the continual maintenance of neighbor connectivity, AODV has a larger overhead than DSR, however, when stressed with high traffic loads or mobility, AODV provides higher throughputs [33]. AODV's use of hop-by-hop routing and a routing table rather than source routing and a route cache means that it conforms to what is usually expected from a routing protocol. The AODV protocol has now evolved into a newer IETF

specification called DYMO; which contains a number of code simplifications that aid implementation. DYMO adopts DSR's path accumulation feature which can reduce the flood of RREQs and RREPs however it is functionally more closely aligned with AODV.

2.4.3 OLSR

OLSR (Optimized Link State Routing) [16] was an was an initial attempt at standardizing a proactive link-state routing protocol. Since the first RFC 3526 [16] in 2003, the conceptual evolution of this protocol has continued with the OLSRv2 [17] draft nearing completion. OLSR has also been extensively implemented. An initial implementation by Tonnesen [15] has been continued by numerous contributors at[3] [24] www.olsr.org and is currently the most used ad hoc routing protocols.

The initial OLSR RFC, 3526 [16], used hop count as a metric, however, problems with this metric surfaced in Tonnesen's initial OLSR implementation [15]. Thus, real world implementations have long since broken conformance with this RFC. OLSRv2 [17] uses the ETX [48] metric for routing.

The characteristic feature of OLSR, which differentiated it from other link state routing protocols, were MPRs (Multi Point Relays). As previously explained, MPRs reduce the number of redundant link state transmissions by electing specific nodes as relays. Selection is performed in a manner such that every OLSR node is a direct neighbor of a MPR. OLSRs MPR function also contains a DPD (Duplicate Packet Detection) function which is used

[3] OLSR contributors: Thomas Lopatic, Hannes Gredler, Bernd Petrovitsch, Aaron Kaplan and Sven-Ola Tücke

to reduce redundant transmission by identifying duplicate packets. Generally the larger and more dense the network, the more optimization can be achieved by using MPRs [17]. In addition to using MPRs to efficiently disseminate information, the OLSR protocol uses FSR (Fish eye State Routing) techniques. These techniques enable frequent updates of nearby nodes and infrequent updates of distant nodes. FSR reduces the overhead of link state messages in larger networks.

Anecdotal criticisms of OLSR state that a significant amount of MPR redundancy is needed to prevent link state databases from becoming desynchronized and forming routing loops. The additional MPR redundancy results in a larger number of redundant updates; reducing performance. These criticisms led others to explore a fundamentally different approach to routing.

2.4.4 BATMAN

BATMAN (Better Approach To Mobile Ad hoc Networking) [56, 19] is a new and different approach to routing. In BATMAN, routing tables are built, hence it is a proactive routing protocol, however, routes are acquired in a biologically inspired manner, sharing similarities with AntHocNET [57].

In link state routing, messages are used to communicate changes across the entire network. Each node then calculates its own SPF tree. In distance vector routing, each node must inform other neighbors of their paths to different destinations, thus every router views the network through the eyes of its neighbors. The BATMAN protocol is fundamentally different. It does not try to discover or calculate routing paths, instead it tries to detect which neighbor offers the best path to each originator [19].

In BATMAN, routing information is not communicated directly, instead, each node broadcasts packets called OGMs (Originator Messages) every second. When received by neighboring nodes, OGMs get re-broadcasted. Route selection for a given destination is based on the node from which the most OGMs have been received for a particular destination. The number of OGMs that can be accepted is limited to a constantly moving window. This window limits the history of which OGMs are allowed to describe a given route.

The scalability of BATMAN counts on packet loss and thus, like other protocols, OGMs are broadcast as unreliable UDP packets. As nodes continuously broadcast OGMs, without packet loss, these messages would overwhelm the network. The scalability of BATMAN depends on packet loss and thus it is unable to operate in reliable wired networks.

This mechanism also means that OGMs from nearby nodes will be frequently received whereas OGMs from distant nodes will be received less frequently. The BATMAN protocol can also use different TTLs in OGMs to limit their dissemination. Both of these functions are similar to the limited dissemination FSR concept [39]. As route selection is based on the number of received OGMs, the metric is ultimately a form of reliability and therefore conceptually similar to ETX [48]; the metric used by both OLSR and Babel.

2.4.5 Babel

Babel [18] is a proactive distance vector routing protocol. Babel is newer than OLSR, BATMAN and HWMP, but interestingly, its design is based on on DSDV [13], the first multi hop ad hoc routing protocol. The sequence number feature, which enables routers to determine the freshness of updates

and thus avoid count to infinity problems, is borrowed from DSDV. Also, the loop avoidance techniques, whereby feasibility conditions must be determined before accepting a route, are adapted from EIGRP [23]. Babel uses ETX [48], and; like other ad hoc routing protocols, updates are transmitted unreliably. Provisions have been made for reliable updates too. IPv6 is used to exchange routing information. The author claims that Babel can outperform other competing routing protocols such as OLSR in sparse networks.

2.4.6 HWMP

HWMP (Hybrid Wireless Mesh Protocol) is a hybrid routing protocol combining both proactive routing and reactive routing techniques. HWMP operates at the data link layer, uses MAC addresses for routing, and the bandwidth aware Air Time Link metric. HWMP can operate in two modes, On-demand HWMP and Tree-based HWMP. On demand HWMP operation is very similar to AODV. In Tree-based HWMP, an additional proactive component is added. This proactive component maintains the shortest path to the root, creating a tree similar to STP (Spanning Tree Protocol). All other routes are discovered reactively. Although the IEEE 802.11s [20] amendment has not been standardized, experimental implementations are available [58].

2.5 Experiment

This experiment compares OLSR, BATMAN and Babel. Attempts were made to use AODV, however like other recent studies [59], problems with the implementaion were encountered. Attempts were also made to use the open

802.11s [58] HWMP routing protocol. Unfortunately, this routing protocol only works with a newer wireless driver known as ath5k. Due to performance problems with this driver, it was necessary to revert to the older MadWiFi driver which ruled out open 802.11s as a consideration.

The BATMAN routing protocol is being developed as both a user-space routing protocol that operates at the network layer as well as a kernel-space implementation running at the data link layer. This study experiments with both routing protocols, referring to them as BATMAN L3 and BATMAN L2. Only a few real world experimental evaluations of these protocols exist [60, 59]. A greater number of experimental tests, from independent sources, are required to ascertain the validity of these conclusions.

In the proposed experiment, the wireless nodes were ALIX 3 500MHz x86 embedded PCs with 256 MB of RAM and Atheros CM9 wireless cards. The platform and routing protocol versions can be found in Table 2.3. It must be noted that all routing protocols were tested with their default configuration. Comparative tests were performed over four different topologies. The first test was performed with all nodes in direct communication range of the gateway and thus no routing should have been occurring. This test served as somewhat of a control. The remaining three tests featured random placement of nodes throughout a building. No specific attempt was made to dictate a particular topology, however, the nodes were placed far enough apart to ensure that multi hopping would be required. In the experimental setup, the transmission power was reduced and all wireless nodes were placed in different rooms. This study measured; packet delivery ratios, bandwidth and routing protocol overheads.

To measure packet delivery ratios, a simple ruby program that sent ICMP

Platform	Version
Voyage Linux	0.6
MadWiFi Driver	0.9.4
Linux Kernel	2.6.30-486-voyage

Routing Protocols	Version
OLSR	0.5.6-rc7
BATMAN-0.3	0.3
BATMAN-ADV	0.2
Babel	0.97

Table 2.3: Platform and routing configuration

messages from the gateway to all nodes was created. The program was written such that only one ICMP message was present in the network at any one time. This ensured that the losses measured did not include congestion based losses. Theses tests were based on the success of 10,000 ICMP messages and were performed many times for each routing protocol in each of the four topologies.

We also performed bandwidth tests. One gateway node was connected to a dedicated server running the lighthttpd web server. Wireless nodes were simultaneously issued instructions to download a large 158MB file from the lighthttpd server. The downloads were timed. The elapsed time between when the download command was issued and the final node completed its file transfer was recorded. The bandwidth results were derived from this time. The bandwidth calculations were based on the transfer of $(7nodes \times 158MB)$ 1106MB of data in the recorded number of seconds. The speed was then calculated $(\frac{1106MB}{time})$ in megabytes per second then converted to megabits per second and graphed. These tests were performed four times for each routing protocol in each topology and the results were averaged.

This study also captured routing protocol overheads. The mesh nodes are not powerful enough to capture the traffic traversing their interfaces when routing thousands of packets per-second which made the determination of

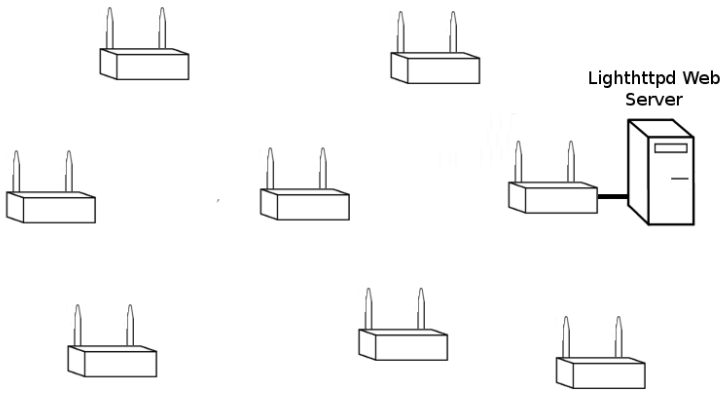


Figure 2.19: Experimental setup

the exact routing overhead difficult. All wireless nodes were placed within range of an external capturing device. Thus, it must be considered that in the scenarios where overheads were measured, every node was in direct communication range of the gateway. Wireshark was used to capture packets over 60 second intervals. Routing traffic was captured both while the network was unloaded and also when the network was loaded with file transfers. An aspect of this test which requires consideration is that the overheads of routing protocols may have been different in topologies where the nodes were not all within transmission range of one another. Unfortunately, the practicalities of measurement made this difficult to accurately measure.

2.6 Results & Discussion

From a qualitative perspective, all routing protocols were equally reliable and rarely suffered from TCP dropouts. Packet delivery ratios for the three routing protocols were also consistent. Results varied for the different

topologies however all packet delivery ratios were between 99.6% and 99.98%. These results differ from other experimental studies which found lower packet delivery ratios [60, 59], however, as the experimental setups were different, direct comparisons are inappropriate.

The results of the bandwidth tests, shown in Figure 2.20, reveal that the Babel routing protocol provided higher throughputs than OLSR, BATMAN L3 or BATMAN L2. Figure 2.20 also shows that BATMAN L2 outperformed BATMAN L3 and OLSR in three of the four topologies, however, the performance differences are too small for definitive conclusions. One peer reviewed study [59] concurs that Babel offers greater throughput than both BATMAN and OLSR. This study also found that OLSR performed poorly, however, discussion following the publication of this paper suggests that this result may have been caused by a bug in the version of OLSR used.

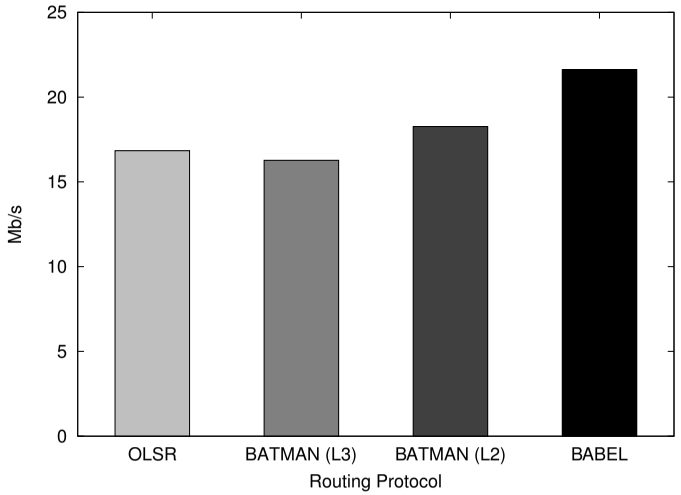
Another study [60] also compared OLSR and BATMAN and found that BATMANs throughput was approximately 15% higher than OLSR. The validity of this result, based on their selection of network variables, is questionable. By default, OLSR has a hello interval of 2 seconds and a topology exchange interval of 5 seconds. Comparatively, BATMAN transmits an entirely different message, known as an OGM, every 1 second. In this study, Johnson et al [60] claims that for fairness reasons, OLSR's hello and topology exchange intervals should be the same as BATMAN's OGM intervals of 1 second. This is an unfair because BATMAN and OLSR are completely different protocols. BATMAN's OGMs are minuscule because they carry very little routing information and are required to be sent more often than OLSR hellos and topology exchanges. Routing protocols should be compared with their default hello and topology exchange intervals because they are the settings

likely to be used by the majority of network administrators.

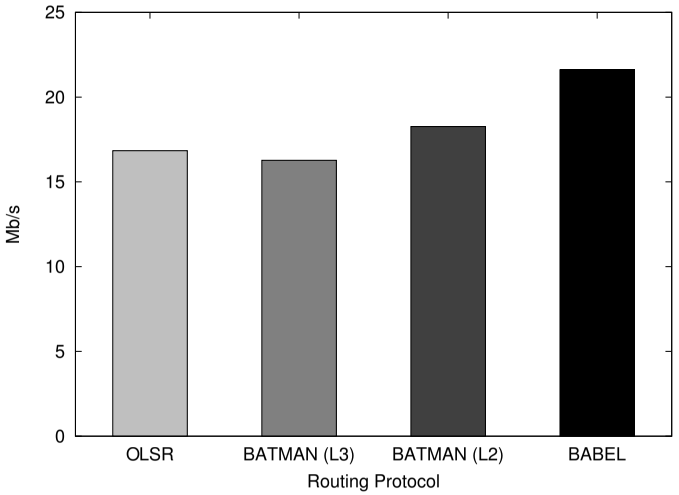
In my study, Babel consistently outperformed other routing protocols. It is questionable whether this is due to a better selection of routes or lower overheads. A curious artifact in the results is seen in topology 1, shown in Figure 2.20. In this topology, no actual routing decisions were being made. Recall that topology 1, was designed as a control and all nodes were within direct range of the gateway. As no routing decisions were being made, throughput differences must be a result of protocol overheads.

The overhead of the routing traffic, in bytes, is shown in 2.21a. This shows that the OLSR routing protocol, under default settings, transfers the most bytes in overheads. Comparatively, Babel produces a minuscule overhead, however, the number of bytes transferred is not a exact measure of overhead. Due to the fixed overheads of IFS (Inter Frame Spacing), DCF (Distributed Coordination Function), preambles and trailers; updates are more efficiently transferred in fewer large packets rather than multiple small packets.

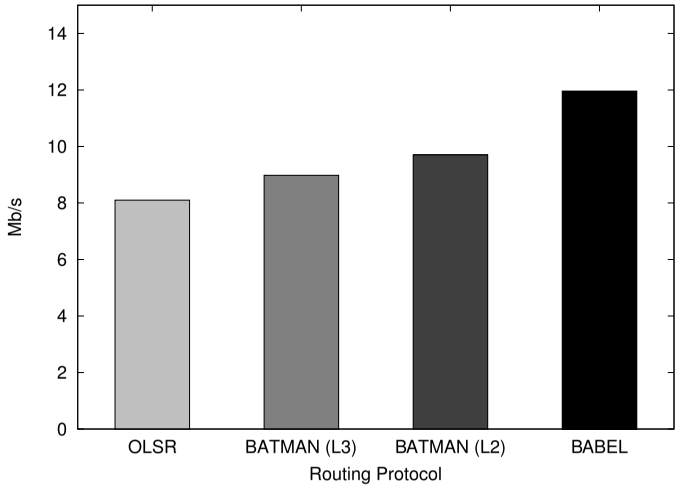
Figure 2.21b shows the number of routing packets transmitted by different routing protocols every 60 seconds. This shows that BATMAN transmits very large numbers of routing packets/frames. Recall that in BATMAN, routes are not exchanged, instead, routes are detected by constantly passing very small messages called OGMs around the network. This functionality describes why BATMAN transmits the largest number of routing messages. It also however, explains why the average packet size is so small. Upon inspection of Figure 2.21c, it is evident that BATMANs routing packets/frames are very small. This is because BATMANs OGMs don't actually carry data about routes. Comparatively, Babel and OLSR messages are much bigger



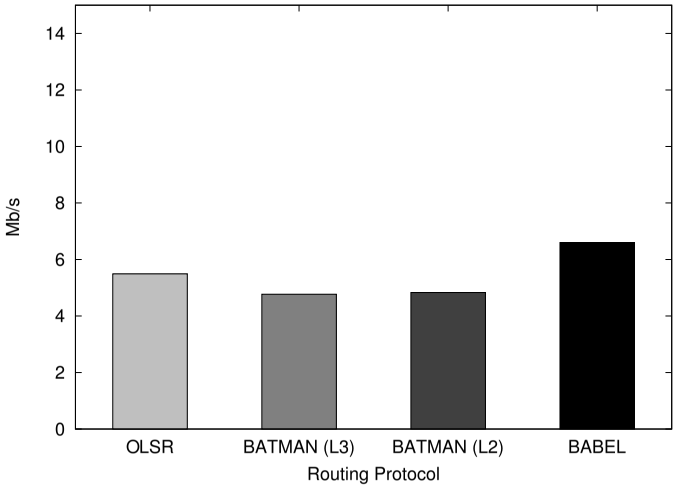
(a) Achieved bandwidth in topology 1



(b) Achieved bandwidth in topology 2



(c) Achieved bandwidth in topology 3



(d) Achieved bandwidth in topology 4

Figure 2.20: Achieved bandwidth

because they have to carry information about routes.

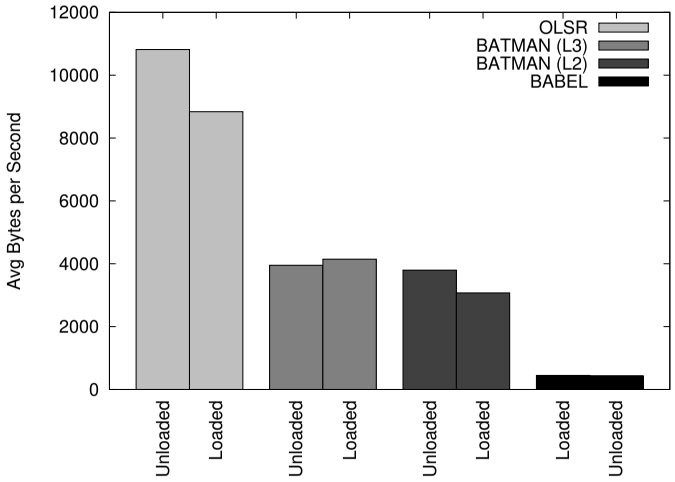
The approximate percentage of channel time consumed by routing updates is calculated using equation 2.4. This calculation includes IFS, DCF preambles, trailers and the size of the packet, divided by the baseline bit-rate. The baseline bit rate is used for all broadcast UDP and TCP segments. Once this value is derived, it can be multiplied by the number of times the routing protocol transmits these packets per second. Our approximation of the channel time used by these routing updates is shown in Figure 2.21d. These results concur with the results of a previous study which suggests that OLSR has a higher network overhead than BATMAN [60].

$$(\frac{AvgPacketSize(inbits)}{1048576}) + DIFS + (preamble + trailer) \tag{2.4}$$

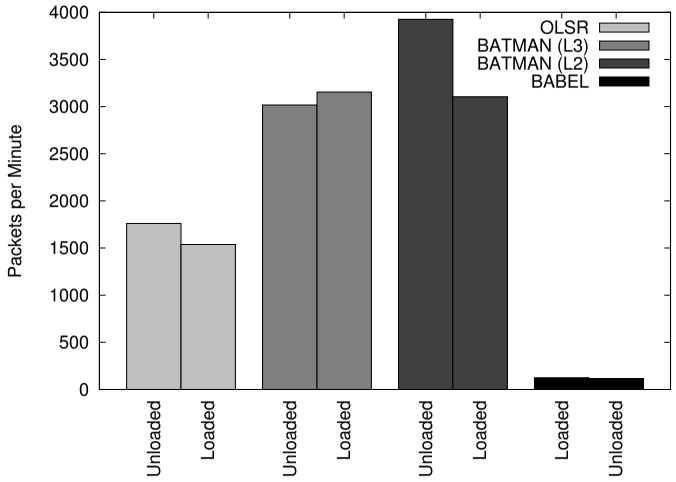
2.6.1 Addressing and layers discussion

This section discusses the trade-offs of performing mesh routing at the data link layer and the network layer. Routing was traditionally envisaged to occur at the network layer and therefore can be used in conjunction with any link layer technology such as Ethernet, ADSL, WiMAX or Bluetooth. This may provide advantages in heterogeneous wireless scenarios. An equally important advantage of routing at the data link layer is that any network layer protocol may operate over the top. For example, IPv4, IPv6 and DHCP will be able to operate and provide convenient addressing mechanisms.

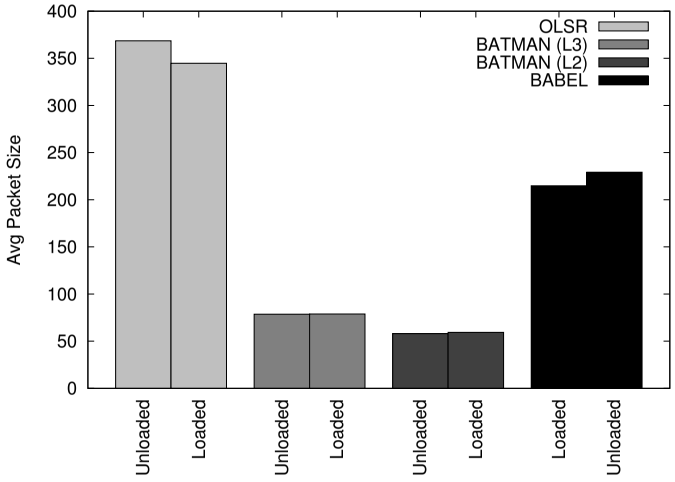
Currently there are a greater number of routing protocols that are imple-



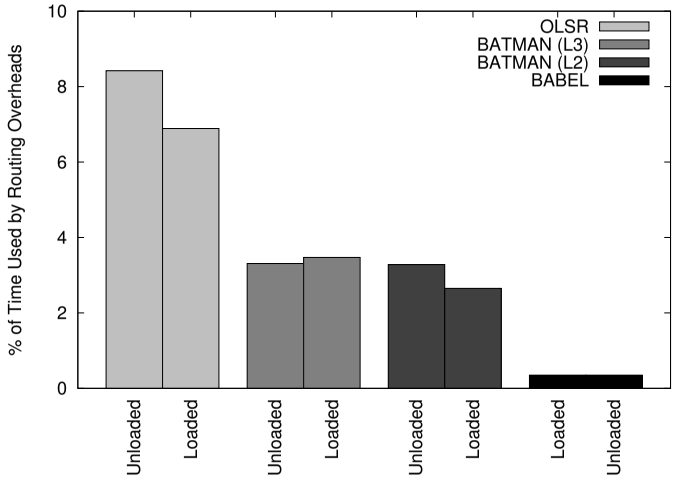
(a) Average overhead in bytes per second



(b) Routing packets transmitted per minute



(c) Average packet size



(d) Percentage of time used by routing updates

Figure 2.21: Routing protocol overhead

mented at the network layer [61, 56, 62, 63] than the more recent data link layer protocols which are comparatively immature [58, 56]. This is currently a debated issue with the IEEE [20], who typically standardize physical and data link layer protocols, and the IETF [4], who typically standardize network and transport layer protocols. Either scheme will violate the layering principle. A flat IP addressing scheme will break the usual hierarchical addressing that occurs at the network layer. Equally, routing at the data link layer is fundamentally wrong because the data link layer will be performing routing; which is a network layer function. Regardless of the layer chosen, traditional layering principles will be distorted.

Upon inspection of the BATMAN L2 and BATMAN L3 results, the only major performance difference between data link layer routing and network layer routing is the packet size. Data link layer routing protocols will not require a network layer IPv4/IPv6 header and may therefore be smaller. This argument is of minimal consequence because the routing protocol is a far bigger determinant of overheads. To illustrate this point, Babel, which uses a large IPv6 header, increasing the packet size of every routing message by 40 bytes, has lower overheads than BATMAN L2 which operates without a IP header. The layer used by the routing protocol has few performance benefits or drawbacks and thus the decision to use one or the other should be architectural.

2.7 Conclusion

This chapter compares routing protocols, however, more specifically, it investigates which features provide the most performance when routing in

mesh networks. This work is original because many comparisons are simulated [11]. Our work is experimental and confirms the findings of the only other peer reviewed experimental study [59] that tested Babel. The conclusion is that in small wireless networks Babel offers higher throughputs. The results also confirm that the overhead of OLSR is higher than BATMAN [60, 59], but contradict other studies that claim large throughput differences between OLSR and BATMAN [60, 59]. The results of this study suggest that the performance of OLSR and BATMAN is similar.

A separate conclusion is that, in small mesh networks, the overhead of the routing protocol has the largest impact on throughput. In the future, I wish to run similar tests in a larger experimental set-up and obtain statistics on CPU load and convergence time. While this study concludes that Babel provides higher throughputs in small networks, it is untested in larger networks. I hope these findings will provide the impetus for further experimentation.

Chapter III

Channel Selection in 802.11 Multi Radio Multi Hop Ad hoc Networks

Despite over a decade of research in multi hop ad hoc networks, a fundamental performance limitation remains largely unsolved. Multi-hop throughput degradations occur when transmissions are hopped over multiple wireless nodes utilizing the same frequency. This limitation is ultimately a contention problem, however, it is exacerbated by numerous ad hoc specific issues. The solution to this problem, which has been attempted at many layers in ISO networking model, is to reduce contention by turning multi hop ad hoc networks from single channel networks into multi channel networks.

This chapter provides an implementation and evaluation of RDCS (Routing Driven Channel Selection), a channel selection mechanism that operates with the OLSR (Optimized Link State Routing) protocol. This cross layer protocol uses information from the routing protocol to make physical layer channel selections in multi radio networks. It circumvents multi-hop performance problems by enabling 802.11 mesh nodes, equipped with multiple radios, to intelligently utilize a range of frequencies, dramatically increasing performance.

3.1 Single Radio Mesh Networks

Performance problems in single radio mesh networks are derived from half duplex communication in 802.11. Single radio nodes are unable to send or receive at the same time. This fundamental problem is many magnitudes worse in a *multi-hop* environment than a traditional *single-hop* AP-client communication paradigm. Early theoretical studies [9, 64, 65] have shown that the throughput of end-to-end data transfers in single-radio *multi-hop* environments quickly diminish with increasing numbers of nodes and hops. Solutions to this problem have been attempted at different layers in the OSI networking model and subsequently, the problem definition varies. This chapter describes the problem from both the data link layer and network layer perspectives which will aid the subsequent reviews of data link and network layer solutions.

3.1.1 Contention

The simplest manner to describe the problem is contention, however, three issues combine to make contention in 802.11 multi hop ad hoc networks especially severe. Firstly, radios are half-duplex, which means that to relay a packet, the radio must first receive and then transmit the packet. As the process of receiving and then transmitting the frame are two autonomous operations, the channel time required to perform this procedure is doubled. If every frame is forwarded in this manner, only half the normal number of packets can be transmitted and subsequently, the capacity is halved.

The second major problem is that, in networks with only a single 802.11

radio, only one channel may be used. In traditional wireless 802.11 networks, nearby APs are deployed on non-overlapping channels to reduce contention. Comparatively, in single radio multi hop ad hoc networks, all nodes must use the same channel to participate. This significantly increases the number of competing transmissions.

The catalyst of these two problems is that the energy from wireless 802.11 transmissions propagate further than the frame recipient [66]. As energy propagates further than the range at which frames can be interpreted, even distant 802.11 nodes must back-off until the frame transmission has finished. For illustrative purposes, Figure 3.1 shows the propagation range of a node at 54Mb/s, 1Mb/s and the interference range. The interference range is the range whereby a receiving radio's carrier sensing mechanism will back-off due to a concurrent transmission. Although the carrier sensing/interference range varies between chipsets, a rough guideline is provided by the network simulators NS-2 and QualNet, which both specify an interference range of twice the transmission range. As a consequence of this large interference range, a single transmission can prevent a large number of nodes from simultaneously transmitting. Theoretically, only every fourth node in a string topology can transmit concurrently [65].

3.1.2 Inter-flow and intra-flow interference

An alternative description of the same problem, often used by authors with solutions operating higher in the OSI networking model, is the problem of interfering flows. Routing metrics and transport layer contributions often break this problem into the sub-problems of *intra-flow* and *inter-flow* inter-

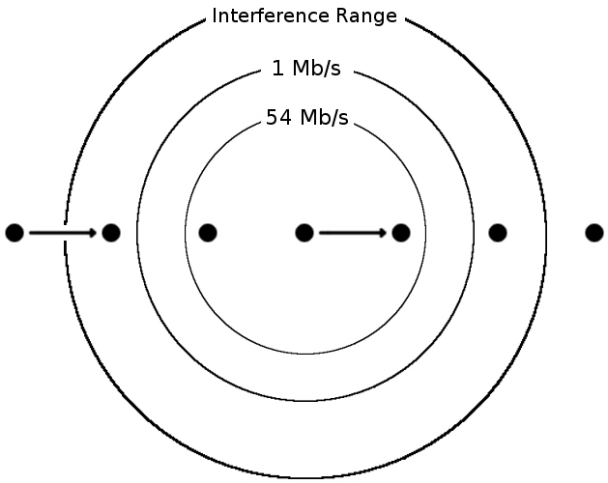


Figure 3.1: Large carrier sensing range

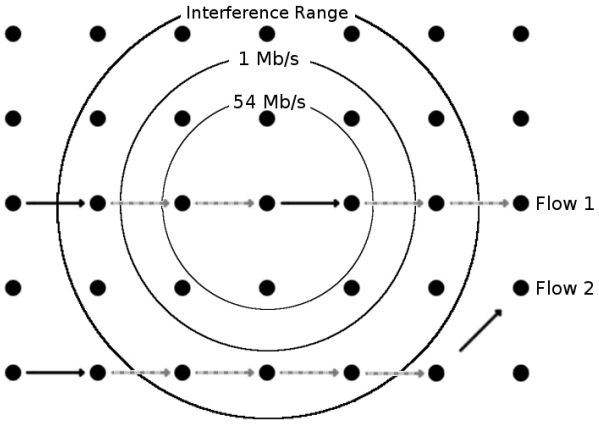


Figure 3.2: Inter-flow and intra-flow interference

ference. *Intra-flow* interference occurs when one node prevents another node from transmitting within the *same flow*. For example, the central node in Figure 3.2 can prevent nearby nodes along the same flow, flow 1, from forwarding concurrently. In contrast, *Inter-flow* interference occurs when two nodes forwarding frames along *different flows* prevent each other transmitting simultaneously. For example, the central node forwarding traffic in flow

1 is preventing a simultaneous transmission in flow 2 (Figure 3.2).

The solutions discussed in this chapter, operate at different layers in the OSI networking model. Despite operational differences, they all attempt to reduce contention by enabling wireless stations to transmit simultaneously on multiple non-overlapping channels. By transmitting frames on different channels, many transmissions can occur concurrently. The goal of this research area is to find an efficient way to change multi-hop ad-hoc networks from single channel networks to multi channel networks. By enabling transmissions over multiple channels, MAC layer or *inter-flow* and *intra-flow* interference problems are reduced because multiple transmissions can be completed concurrently. Considering the severity of throughput degradation in single radio multi-hop ad-hoc networks, this is a highly important problem.

3.1.3 Performance of multi hop single-radio networks

Gupta and Kumar's paper [9] demonstrated that for a given number of nodes (n), each of which is communicating with another node, the maximum possible throughput, considering *optimal node placement* and communication pattern, is $\theta(\frac{1}{\sqrt{n}})$. Furthermore, it was also shown that, given a *random node placement* and communication pattern, the throughput available to each node follows the function $\varpi(\frac{1}{n\sqrt{logn}})$ [9]. These degradations in throughput were confirmed by Li et al [65] who used the combined methods of simulation and practical experimentation. Currently, results suggest that, as the number of nodes increase, the bandwidth available to individual nodes approaches zero.

As stated in the problem description above, the large interference range of 802.11 wireless mesh networks means that only every fourth node in a chain

can operate simultaneously. In practical and simulated studies, Li et al [65] found that the random contention mechanism of 802.11 led to poor scheduling across a chain of nodes. Put simply, as TCP data and ack segments travel along a string of ad hoc nodes, 802.11 schedules the transmission of these segments randomly based on DCF (Distributed Coordination Function). The conclusion was that poor scheduling of transmissions meant that only one in every seven nodes in a wireless chain could transmit concurrently. This means that the real world scalability of multi hop ad hoc networks is significantly worse than the most optimistic theoretical estimates.

To enable mesh networks to increase in size and provide greater bandwidth, wireless nodes must utilize multiple frequencies. If nodes could utilize orthogonal frequencies they could dramatically reduce contention because simultaneous transmission can now occur on different channels. By reducing contention in this manner, performance increases are possible. This review of prior work begins at the MAC layer.

3.2 Multi-Channel MACs

To alleviate contention problems in single channel multi hop networks, a number of MC-MAC (Multi Channel MAC) protocols have been proposed. New MACs designed for multi channel multi hop networks aim to spread transmissions over a large number of non-overlapping channels. By splitting the transmissions across multiple channels, contention is reduced because multiple simultaneous transmissions may be performed over different channels. A number of different MC-MACs have been proposed. This chapter separates the schemes into different categories: multi-radio MC-MACs,

single-radio MC-MACs and hybrid approaches.

3.2.1 Multi-radio multi-channel MACs

The multi-radio MC-MAC protocols [67, 68, 69] require two or more radios to operate. These approaches use one radio for control traffic (such as management frames) and the other radio for data transmission which can occur over multiple different channels. The control radio is used to negotiate and setup data transmissions which will occur later on separate data channels. As multiple channels are available, data transmissions may occur simultaneously on different channels.

Wu et al [68] introduced one of the first MC-MAC protocols. Using this MAC, a node wanting to transmit begins by sending a Request-to-Send (RTS) message over the common control channel. Included in this RTS is a Free Channel List which contains a list of channels that the sender can currently use on its alternative radio for data transmission. The receiver replies over the control channel with a Clear-To-Send (CTS) confirming the channel to be used for data transmission. This mechanism of scheduling transmissions over a default control channel is shown in Figure 3.3. A different paper by Jain et al [69] operates in a very similar manner whereby the receiver chooses the data channel based on a Free Channel List sent by the transmitter.

Kyasanur et al [70] builds upon previous work [68, 69], introducing a number of new ideas. The authors suggest that the control channel should be a slice of unlicensed spectrum in the 900MHz ISM band. By utilizing the 900MHz band, greater range can be achieved over the 2.4GHz or 5GHz band. It is claimed that the greater range may reduce hidden terminal issues.

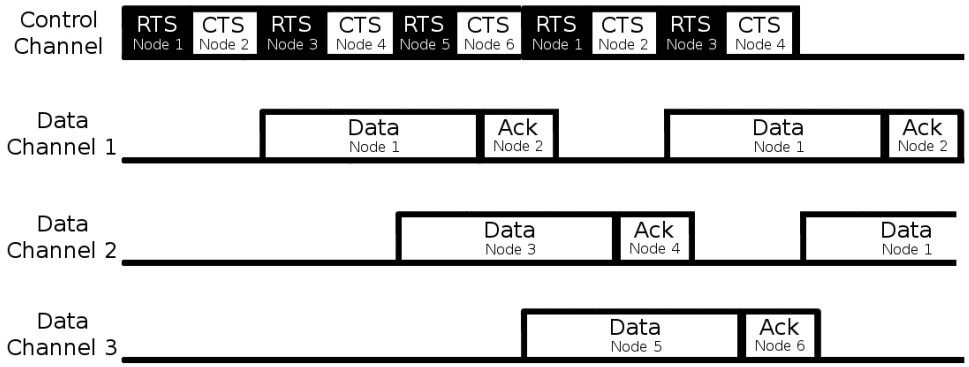


Figure 3.3: Multi-radio multi channel MAC protocols

However, the 1-2MHz slices available in the 900MHz band provide lower speeds than the larger 20MHz channels in the 2.4GHz and 5GHz band. As a result, the control channel becomes a bottleneck. To overcome this problem, mechanisms that enable multiple packets to be sent per RTS/CTS exchange are devised. To further increase the efficiency of the control channel, the RTS/CTS reservation mechanism may also schedule packets in advance. The authors [70] claim that by using advanced scheduling in the control channel, greater channel utilization can be achieved.

While the ideas in Kyasanur et al's [70] study are meritorious, there are a number of practicalities that impede implementation. This proposed MAC is untested in a real world environment. Thus, clustered packet transmissions and advanced scheduling may have unknown affects on TCP. Furthermore, it may be more practical to use the 2.4GHz band rather than the 900MHz band for the control channel because equipment is mass produced, cheap, and high bandwidth which prevents the previously discussed bottleneck issues.

The multi radio MC-MAC approaches [67, 68, 69] can potentially increase the available bandwidth in both AP configurations and multi-hop ad-hoc

networks through better spatial reuse. A disadvantage is the cost of an additional radio in terms of price, power consumption and the physical size of the network device. Furthermore, the efficiency of using an entire channel for control messages is also debatable. Questions over the acceptability of such trade-offs led to the development of single radio MC-MACs.

3.2.2 Single-radio multi-channel MACs

The single radio multi channel MAC approaches only require one interface and can transmit on multiple different channels. They facilitate transmissions on different channels by requiring every node to be time synchronized. A couple of different approaches [71, 72] have been proposed.

In SSCH (Seeded Slotted Channel Hopping) [71], each wireless radio hops in a predetermined pattern through a list of channels, remaining on each slot or channel for 10ms. While each node has a different hopping pattern, hopping sequences are designed such that different nodes will settle on the same channel at least once per iteration of the hopping sequence. When a node has data that needs to be transmitted to a specific node, it must wait until the two nodes hopping sequences overlap and both radios settle on the same channel. To ensure that hopping sequences do overlap, there is a parity slot once every iteration whereby all radios converge on one channel.

Another approach by So and Vaidya [72] works in a very similar manner to the previously discussed multi-radio MC-MACs because the MAC negotiates the preferred channel via RTS/CTS messaging. However, the single channel MC-MACs are constrained compared to their multi-radio counterparts as they may only operate on one channel at any one time. So and Vaidya [72]

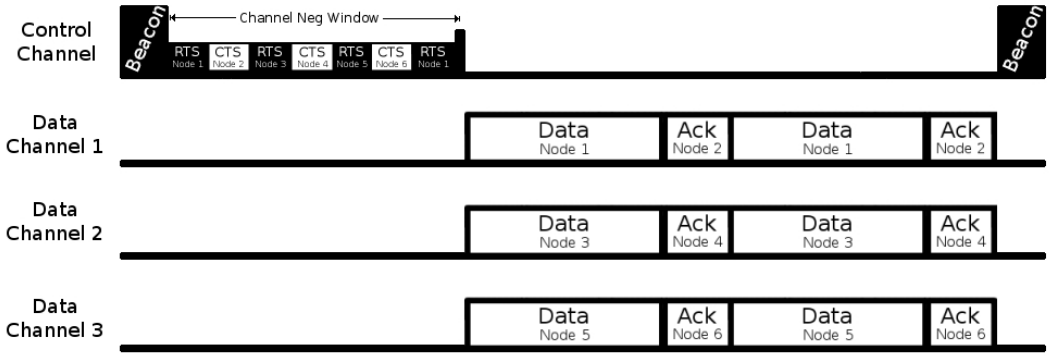


Figure 3.4: Single-radio multi channel MAC protocols

use a window, during which, all nodes must simultaneously switch their radio to the control channel, to negotiate future transmissions. By introducing a channel negotiation window, nodes will capture all RTS/CTS exchanges and therefore will be aware of all future data transmissions and the channel on which they will occur. Figure 3.4 shows So and Vaidya's [72] approach whereby after every beacon, a channel negotiation window is opened. Following this window, the negotiated data transmissions are sent over several different data channels. This approach [72] was similar to a component of the 802.11s draft called CCF (Common Channel Framework) [20]. However, since the release of this draft [20], CCF has been removed from 802.11s due to interoperability concerns with existing 802.11 DCF [3] functions.

The single radio multi channel MACs [71, 72] accomplish transmissions across multiple different channels with a single radio. The difference between these schemes is that SSCH (Seeded Slotted Channel Hopping) [71] hops in a semi scheduled manner through channels whereas the previously mentioned approach by So and Vaidya [72], negotiates data transmissions over a common control channel. By utilizing multiple different channels simultaneously these

schemes can potentially increase the aggregate throughput, however, there are a number of inefficiencies or drawbacks.

Firstly, there are periods of time where no data can be transmitted. In SSCH [71] this occurs when hopping across channels where there is no data to be transmitted. A given node may only spend a tenth of its time on a channel with the node it wishes to communicate with. This is also prevalent in So and Vaidya's approach [72] because, during the channel negotiation window, data is unable to be transmitted. In both of these approaches, nodes may have data that must be en-queued until the two hopping paths overlap or can be negotiated again.

Another problem with these schemes is that they require strict time synchronization to operate. In SSCH [71], hopping sequences must be synchronized so that nodes fall on the same channel at the same time. Equally, in [72], all nodes must simultaneously switch to the common control channel for RTS/CTS exchanges. The these requirements create additional complexity given the known time synchronization issues in large ad-hoc 802.11 networks [73].

3.2.3 Hybrid

The paper by Kyasanur et al [67] is difficult to categorize. While it does not define a new MAC layer, the switchable interface strategy is very similar to the MC-MAC approaches. This hybrid approach requires each node to be equipped with three interfaces. One of the three interfaces is a fixed static interface; used purely to receive traffic. Figure 3.5 shows how the other two interfaces are switchable and used for transmission. If node 2 wishes to send

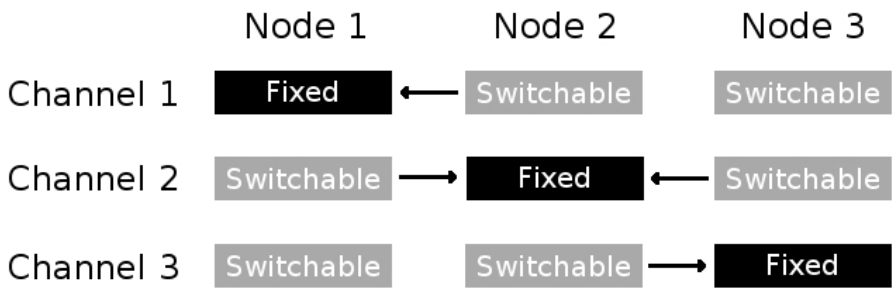


Figure 3.5: Hybrid approach

a packet to node 1, it must switch one of its switchable interfaces to the static channel of node 1. This approach requires each node to build and maintain a list of its neighbors and the channel used by the static radio. This hybrid protocol operates at the data link layer and subsequently any routing protocol may be used on top of it. One disadvantage of this approach is the large number of radios required for operation. The other problems are common for all the MC-MACs and will be discussed jointly.

3.2.4 Multi-channel MAC criticisms

There are a number of criticisms relevant to multi-radio MC-MACs and single-radio MC-MACs that have prevented practical usable implementations. Firstly, the support of broadcasts is inefficient. In multi channel 802.11 networks, broadcasts must reach all nodes. The lack of a common data channel requires broadcasts to be retransmitted by each node over every channel.

Another problem is the channel switch delay. The channel switch delay is the hardware and software delay that results when a wireless card changes

channel or switches between frequencies. It is a null period during which the interface is unable to send or receive frames. Because many MC-MAC approaches switch channels on a per-packet basis, the channel switch delay is troubling. Some studies claim that actual hardware switching delays are 80μs [71]. Other studies [74] found much longer delays of 5ms and 20ms in Atheros and Intersil Prism based wireless cards. Shin et al [75] found that channel switch delays in Intersil Prism wireless cards were 22ms. My studies of the switching delay [76, 77] found delays between 2ms and 22ms varying with the chip-set. The performance of MC-MAC approaches depended on significantly lower channel switching delays in future 802.11 equipment.

The lack of open source wireless drivers is also problematic because it prevents researchers from experimenting with new MAC layers. It is hoped that recent efforts such as the ath5k driver [78] may create opportunities for actual implementations of these MACs.

The final, and perhaps most powerful hindrance to these schemes is interoperability. When the IEEE 802.11s working group was in its preliminary stages, a single radio MC-MAC scheme similar to So et al's [72] was proposed. However, this feature was ultimately dropped due to concerns over interoperability with existing 802.11 Distributed Coordination Function (DCF) functions. This interoperability problem is generalizable to all of the MC-MACs. It prevents the advantages of 802.11, such as low cost and continual research, from being applicable for multi hop ad hoc networks.

3.3 Channel Aware Routing Metrics

This contention problem in multi hop ad hoc networks can also be solved

at higher layers. By solving the problem at a higher layer and using the existing 802.11 ad hoc mode, 802.11 compliance can be maintained. To solve this problem at higher layers, multiple radios are required per node. In addition, these nodes also require a mechanism to semi-permanently assign channels to these radios to create a channel diverse topology. Furthermore, the routing protocol must be engineered to prefer channel diverse paths.

Routing metrics were previously discussed in chapter 2, however, in this section, they are analyzed with specific reference to minimizing inter-flow and intra-flow interference. Some studies [53] have found significant performance gains using dual radio nodes, an identical channel assignment for all nodes, and an intelligent channel aware routing metric. Thus, despite a marginal improvement in channel diversity, an intelligent channel aware metric was able to make significant use of the spatial diversity. Our discussion of routing metrics begins by re-reviewing the ETX (Expected Transmission Count) and ETT (Expected Transmission Time) metrics. While neither of these metrics are channel aware, they form the basis of more complex channel aware metrics that are capable of representing the effects of inter-flow and intra flow interference.

3.3.1 ETX and ETT

ETX is a reliability metric. It measures the number of transmissions required to successfully move a packet from the source to the destination. ETX [48] is superior to the hop count metric and is currently used in the OLSR and Babel routing protocol. ETX was improved by Draves et al [53] who extended it by adding a bandwidth measure. The resulting metric is called

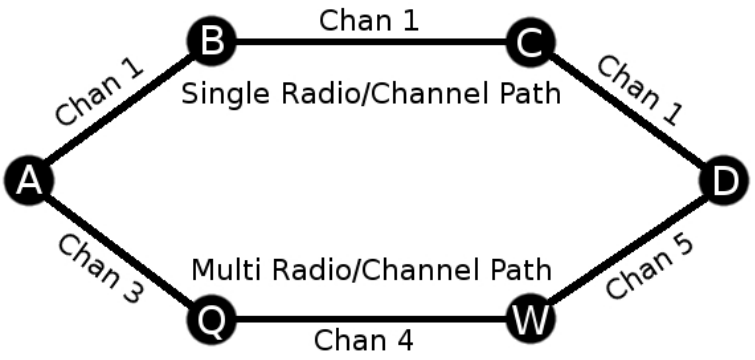


Figure 3.6: ETT/ETX and intra-flow interference

ETT [53] (Expected Transmission Time), however, neither ETX or ETT incorporate the detrimental effects of inter-flow or intra-flow interference. This limitation is demonstrated in Figure 3.6 where node A, using ETT or ETX, is unable to differentiate between the two paths from A to D; despite the fact that route ABCD will provide significantly less throughput than AQWD. A more in-depth discussion of ETT and ETX was provided in section 2.3.5.

3.3.2 WCETT (Weighted Cumulative Expected Transmission Time)

WCETT (Weighted Cumulative Expected Transmission Time) [53] improved upon ETT by capturing the effects of intra-flow interference. *Intra-flow* interference is caused by two or more nodes which degrade the performance of the path by using the *same channel* along the *same flow*. Simply, WCETT punishes paths which utilize the same channel over subsequent links. WCETT, shown in equation 3.1, is best understood in two halves. The first half, $\sum_{i=1}^{n} ETT_i$ is the sum or total cost of ETTs on a path. The second half of the metric, $\max_{1 \leq j \leq k} X_j$, counts the number of times each channel is used along a path. By counting the maximum number of times a channel is

used along a path and adding it to the cost, single channel paths, such as ABCD in Figure 3.6, are punished. The balance between the two components, $\sum_{i=1}^{n} ETT_i$ and $\max_{1 \leq j \leq k} X_j$, is determined by the tunable parameter β. WCETT improves upon ETT by capturing intra-flow interference, however, it loses a property called isotonicity, which will be explained in more detail in section 3.3.4. Furthermore, although it captures intra-flow interference, it does not capture inter-flow interference.

$$WCETT_p = (1 - \beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \max_{1 \leq j \leq k} X_j \tag{3.1}$$

3.3.3 MIC (Metric of Interference and Channel-Switching)

MIC (Metric of Interference and Channel-switching) was proposed by Yang et al [79] and improves upon WCETT by also capturing the cost of inter-flow interference. *Inter-flow* interference is caused by two or more nodes which degrade each others performance by using the *same channel* on two *different paths*. The formula used to calculate MIC is shown in equation 3.2 where N is the total number of nodes in the network and min(ETT) is the lowest ETT on the path. The other two components are IRU (Interference Resource Usage) and CSC (Channel Switching Cost). IRU measures the *inter-flow* interference and CSC measures the *intra-flow* interference. IRU is the ETT of a given link multiplied by the number of neighboring nodes (N_l) sharing the link (see equation 3.3). Higher numbers of neighbors sharing the channel result in a higher metric. The CSC, shown in equation 3.4,

gives different weights (w_1 or w_2) depending on whether the previous channel ($CHprev(i)$) is the same as the current channel ($CH(i)$). A higher weight will obviously apply to paths using the same channel consecutively. Similar to WCETT, MIC is a non-isotonic metric.

$$MIC(p) = \frac{1}{N \times min(ETT)} \sum_{link\, l \in p} IRU_l + \sum_{node \in p} CSC_i \tag{3.2}$$

$$IRU_l = ETT_l \times N_l \tag{3.3}$$

$$CSC_l = \begin{cases} w_1 & if\ CH(prev(i)) \neq CH(i) \\ w_2 & if\ CH(prev(i)) = CH(i) \end{cases} \tag{3.4}$$

3.3.4 Isotonicity and link state routing

The WCETT and MIC metrics are both capable of capturing channel information, but, they are non-isotonic. A metric is said to be isotonic if it preserves the relative link weight of a path when prepended by a common path [79]. Yang et al [79] show why WCETT is non-isotonic and therefore efficient algorithms, such as Dijkstra's algorithm, cannot be used to compute shortest paths.

This section provides an equivalent example. Given the topology shown in Figure 3.7, this example will show how WCETT chooses a suboptimal path to node Z causing routing loops. Paths are calculated using the formula of

WCETT shown in the footnote[1]. It is assumed that a value of 0.5 is used for β. For the upcoming example, the interested reader should follow the WCETT calculations referenced in the footnotes.

When node A calculates a route to node E it chooses the lower cost path ATE rather than ASE[2]. Because of the way the Dijkstra algorithm works, the path ASE will subsequently be removed from further consideration. When node A wishes to calculate the lowest cost path to node Z, it must now chose between ATEZ and ABGNZ. As WCETT punishes paths that use the same channel, over multiple hops, path ABGNZ is the shortest path[3]. However, this calculation is actually incorrect. The shortest cost path is actually ASEZ[4] but this will never be calculated because, following the calculation of path ATE, ASE was discarded as a possible path to node E.

This causes major problems when node B attempts to determine its lowest cost path to node Z. Unlike node A, when node B determines its best path to node E it will choose BASE rather than BATE[5]. This change occurs because the link between node B and node A uses the same channel as the

[1] $WCETT_p = (1 - \beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \max_{1 \leq j \leq k} X_j$
$WCETT_p = (0.5 \times \sum_{i=1}^{n} ETT_i) + (0.5 \times \max_{1 \leq j \leq k} X_j)$

[2] ATE= $(0.5 \times 0.4) + (0.5 \times 1)$
ATE $= 0.7$
ASE= $(0.5 \times 0.5) + (0.5 \times 1)$
ASE $= 0.75$

[3] ATEZ $= (0.5 \times 0.6) + (0.5 \times 2)$
ATEZ $= 1.3$
ABGNZ= $(0.5 \times 1.3) + (0.5 \times 1)$
ABGNZ $= 1.15$

[4] ASEZ $= (0.5 \times 0.7) + (0.5 \times 1)$
ASEZ $= 0.85$

[5] BASE $= (0.5 \times 0.6) + (0.5 \times 1)$
BASE $= 0.8$
BATE $= (0.5 \times 0.5) + (0.5 \times 2)$
BATE $= 1.25$

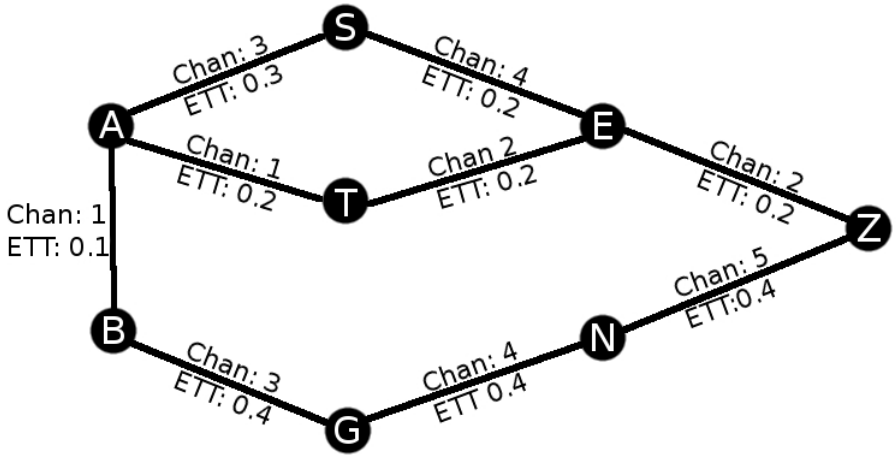


Figure 3.7: Sample topology for WCETT metric calculation

link between node A and node T, and subsequently, that path is penalized. When node B wants to determine its lowest cost path to node Z, it has a choice between BASEZ and BGNZ[6]. Using the WCETT metric, Dijkstra will chose the route BASEZ as the lowest cost path from node B to node Z. This causes a routing loop because node A has calculated the lowest cost path via node B and node B has calculated the lowest cost path via node A.

The problem with WCETT and MIC is that the cost of a link can be changed by links that precede or succeed the measured path. In the case of the path ATE, the preceding link, BA, or the succeeding path, EZ, will modify the cost in a manner that is not relative to the added weight of the additional links. Put simply, the added weight of EZ will differ based on whether it is appended to ATE or ASE. The solution to the problem of non-isotonic metrics is non-trivial.

[6] BASEZ $= (0.5 \times 0.8) + (0.5 \times 1)$
BASEZ $= 0.9$
BGNZ $= (0.5 \times 1.2) + (0.5 \times 1)$
BGNZ $= 1.1$

The authors of MIC [79] have a method to transform MIC into a isotonic metric by virtualizing interfaces into nodes. They simulated this mechanism in [80] with a specially created routing protocol called LIBRA, however real world implementations are non-existent. Currently, link state routing protocols are unable to use metrics that capture the effects of channels or intra-flow and inter-flow interference.

Identical static channel assignments have been used thus far to obtain results for new routing metrics [53]. Due to problems with non-isotonic routing metrics and link state routing protocols, the results have largely been obtained using on-demand routing protocols such as LQSR or AODV [49, 81]. The positive results suggest that channel aware metrics are capable of making better use of the available bandwidth. However, the potential benefits of using a channel aware routing protocol are limited if an identical channel assignment is used on every node. If an intelligent, channel diverse assignment mechanism could dynamically assign channels to radios in a manner that preserves connectivity, throughput improvements could be many magnitudes greater [64]. Cross layer channel assignment attempts to provide a channel diverse assignment. Maximizing the number of collision domains and enabling more concurrent transmissions could provide greater throughput.

3.4 Prior Cross Layer Channel Assignment

Channel assignment in multi radio networks is a trade-off between interference and connectivity. Maximum connectivity can be provided by assigning the same set of channels to every node. However, this will cause large amounts of interference/contention. Another equally simplistic solution is to

assign a random channel to every radio. This will minimize the interference, however, it is unlikely that the network will be completely connected because nodes may only communicate if they share a channel and are within transmission range. The key to channel assignment is to guarantee connectivity while simultaneously minimizing the number of interfering links. Prior approaches can be divided into centralized approaches, where there is a single central channel assignment server, and distributed approaches where each node makes decisions based on locally available information.

3.4.1 Centralized

The centralized approaches [82, 83, 84, 64, 85] rely on a centralized server to assign channels to nodes in the network. The graph based solutions [82, 83, 84] often formulate the problem in graph theoretic terms. They begin with the assumption that an input to their algorithm is the unit disk graph or $G = (V, E)$ where the graph G has V (vertices) representing nodes and E (edges) representing possible links. Most of the channel assignment schemes use variations on graph coloring algorithms such as weighted conflict graphs. The goal is to assign channels, (colors) to links, (edges) in such a manner that maintains connectivity yet simultaneously minimizes the number of nodes sharing the same channel [82, 83, 84]. For a given graph, the centralized algorithm searches for solutions whereby certain network variables such as interference are minimized. However, this has been shown to be NP-hard [64, 86, 84, 83]. As the complexity of finding a solution is exponential, to enable these algorithms to terminate in polynomial time, heuristic/approximate approaches are used.

Other centralized channel selection approaches formulate the problem based on network flows [85, 64]. These channel assignment schemes are similar because their algorithms assume that the unit disk graph is an algorithm input, however, they differ because the aim is not to minimize interference but maximize the number of simultaneous network flows [85, 64]. The centralized network flow schemes maximize the end-to-end bandwidth in the network. In addition, because of the circular dependency between channel assignment and routing, the problem is solved jointly. End-to-end bandwidth is maximized based on assumed knowledge of the traffic profile and the bandwidth that each potential link can offer in advance. To surmise, in addition to assuming knowledge of the entire network as a graph, these algorithms also assume the traffic profile and the bandwidth of every potential link.

The highly theoretical nature of these studies has meant that only a couple of small implementations have been attempted. Two studies [64, 84] provided implementations with 6 and 4 nodes respectively. In these implementations all radios use channels in the 2.4 GHz spectrum which, due to the small number of channels, ensures a fully connected mesh and makes it difficult to conclusively show that such centralized schemes are a practical solution to the channel assignment problem.

3.4.1.1 Criticism

The theoretical nature and lack of experimentation with centralized approaches has disregarded a number of important issues. The unit disk graph, which is an assumed input for many of the previously mentioned channel assignment algorithms, is difficult to obtain and can vary with time. A couple

of papers have been published in topology discovery [87, 88], however, to my knowledge, there is no practical method to obtain complete knowledge of a graph. In addition to the impracticalities of obtaining a graph of the entire network, the network flows papers [85, 64], further assume that potential link bandwidths of edges in their graph are known in advance. In reality, the traffic profile will be highly dynamic and is likely to vary with time.

Another question that centralized approaches must answer is; how changing link conditions be accommodated. For example, if the network is partitioned, how does a centralized approach rectify this problem. One solution to this problem is to have a common control channel [84] which will ensure that the network is never partitioned. However, the control channel will quickly become a bottleneck as the size of the network expands. Section 3.5.2 discusses a phenomenon called inter-radio interference which shows that each mesh node is, in a practical sense, limited to a finite number of radios. Given the finite number of radios that may be used per node, a control channel is an inefficient use of a radio. To conclude, many of these approaches assume a controlling autonomous entity with simultaneous knowledge and control over every node in the network which is impractical in wireless mesh networks.

This section has discussed why the algorithm inputs are unable to be captured and why disconnected nodes in a partitioned mesh may be unable to contact the centralized channel assignment server without a omnipotent controlling entity. However, there are also a number of practical implementation scenarios where a centralized approach is not applicable. In community/neighborhood networks it is unlikely that a centralized server with total control will be a viable approach. A centralized assignment algorithm also introduces a single point of failure and a point of attack for DoS. Furthermore,

it is unknown how such centralized schemes will scale to large and complex mesh networks. Subsequently, they may not provide the self-configuring, self-healing properties originally envisaged for wireless mesh networks.

3.4.2 Distributed

Recent publications [89, 90, 86] suggest a move towards distributed channel assignment mechanisms. The distributed approaches often propose a physical architecture and experiment with their implementation in a testbed. A few different approaches have been proposed [89, 90, 86], however, they are all relatively unique and will be reviewed individually.

An approach by Raniwala et al, known as Hyacinth [90], is designed to provision Internet access across the mesh. The approach is similar to 802.1D Spanning Tree Protocol (STP) because the network formed is a tree structure spanning away from the gateway. Each node contains two or more NIC's, including an up-NIC and a down-NIC. A nodes up-NIC connects to the upstream node's down-NIC which offers the lowest cost path to the gateway. Node's down-NICs do not explicitly connect to another node, but instead select a channel with little or no interference. Routing is simplified as all traffic is routed upstream towards the gateway. Hyacinth aims to provide the fastest possible path to a Internet gateway. Each node offering Internet access is equivalent to a root node in STP whereby a tree is formed spanning away from the gateway. The Hyacinth definition of connectivity is relaxed compared to other approaches as they assume that mesh connectivity will be provided if every node can reach the gateway. Hyacinth [90] was tested in a 9 node prototype. The main criticisms of Hyacinth is that they use an

antiquated STP design. This will be further explored after a description of Dmesh.

Dmesh [86] is similar to Hyacinth as it is also a gateway centric approach, which builds trees spanning away from the gateway. Dmesh is unique because it is based around directional antennas. In the Dmesh design [86], each node has a minimum of one omni directional antenna and one directional antenna. The omni directional antenna is used for locating new networks and also for fault tolerance. Once a network has been found, directional antennas with a 45 degree beam width are used to connect the child node to the parent node. These antennas need manual steering on both the parent and child node. After the directional antennas have been manually aligned, a channel selection mechanism will assign an appropriate channel. The selection is based on the geometric information of surrounding nodes, the direction of the antennas and also locally obtainable information such as interference received by the radio. Dmesh uses a modified version of the OLSR routing protocol [15] to enable it to work in this manner. Extensive testing was performed on a 16 node prototype network. The main drawback of Dmesh is that the directional antennas on both the parent and child node require manual steering by an administrator. This manual configuration somewhat violates the original requirements of self-forming and self-configuring mesh networks.

A further criticism, applicable to both Dmesh and Hyacinth is that the use of a STP/tree topology is inferior to a fully routed network. A STP topology will concatenate paths and flows reducing the available bandwidth. Traffic that is not destined for the gateway, may have to travel upstream when a more direct and less congested path may exist elsewhere. Another problem

with a STP design is the inability to multi-home or operate in the presence of multiple gateways. If multiple gateways do exist, each node can only be part of one tree. The authors [90] claim that 100% of the traffic is Internet traffic, however, this is unrealistic and may neglect other applications such as telephony, gaming and file-sharing that may not travel via an Internet gateway.

In contrast to the gateway centered approaches, Ko et al [89] designed a channel selection mechanism that can support multiple gateways, does not build a STP tree, and ensures mesh connectivity. For ease of exposition, this approach is referred to as DCA, an acronym for the papers title *Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks.* DCA is based on a two radio architecture with one radio operating on a default common control channel and the other radio being used to provide channel diversity. A 5GHz radio is used for the common control channel while the other 2.4GHz radio may operate on a number of different channels. The 2.4GHz radio switches between channels in the 2.4GHz spectrum with the goal of individually lowering the local interference. The paper includes a proof showing that greedily lowering the local interference also lowers the global interference. DCA was tested in a 14 node testbed with the Microsoft LQSR routing protocol and the WCETT metric.

DCA [89] is different from previous approaches [86, 90] and even RDCS, as it operates independently of the routing protocol. This unique advantage enables it to work with any routing protocol. However, DCA only allows a finite number of channel changes. This is designed to allow the channel assignment to stabilize, however, this may not allow the network to adapt to future topology changes. In the authors defense, they claim that chan-

nel selection should be strongly based around the physical structure of the network and that the routing protocols should react to topology changes. Also, the use of a common control channel ensures global mesh connectivity, however, the penalty of a common control channel is that the potential for channel diversity is restricted. It has been argued [90] that the use of a common control channel is inefficient because the number of radios that can simultaneously operate on one node is limited.

3.5 Channel Policy & Architecture

3.5.1 Frequency availability

When 802.11 was first released in 1999, it was designed for operation in the 2.4GHz and 5GHz band. The unlicensed frequency bands differ for each country. The frequency range available in Australia, which closely follows the USA, is shown in Table 3.1. Although there are 11 channels in the 2.4GHz spectrum, the channel separation between these channels is so small that only channels 1, 6 and 11 can be simultaneously used. Compared to the 2.4GHz band, the 5GHz band is significantly larger. However, 5GHz 802.11a wireless equipment has thus far been unpopular. The reasons for this include, slow time to market, poor range in indoor environments and higher cost. Some of these problems, such as time to market and cost, were a product of inconsistent global frequency regulations for the 5GHz band which, for a number of years, prevented 802.11a equipment being sold on global scale. As a result, the majority of 802.11 equipment operates in the 2.4GHz spectrum. For mesh networks, 5GHz 802.11a equipment is becoming increasingly important

Table 3.1: Available frequency for 802.11 use

Band	Frequency Range	Available Channels
ISM	2.400 - 2.475 GHz	3
UNI 1	5.18 - 5.32 GHz	8
UNI 2	5.745 - 5.825 GHz	5

because of the need for multiple non-overlapping channels. Also, a newly discovered phenomenon known as inter-radio interference further increases the need for the frequency diversity and separation that the 5GHz spectrum can provide.

3.5.2 Inter-Radio interference

During experiments, numerous research efforts [64, 91, 53, 92] have encountered significant interference resulting in throughput problems from two closely located radios. This problem of inter-radio interference is prevalent even when the two radios are using non-overlapping channels. From a design and architecture perspective, the small form factor of mesh nodes will require multiple radios to be closely located. The effect of inter-radio interference was highlighted in an experiment by Raniwala et al [64]. Their results have been replicated in in Table 3.2. In this experiment, two 802.11b cards in the same machine were operating on channel 1 and 11 in the 2.4GHz band. The results are presented as a percentage of the throughput provided by the radios given their action: send, receive or silent. Despite using channel 1 and channel 11, which are completely orthogonal channels, the results show a substantial drop when both channels are sending and/or receiving simulta-

Table 3.2: Inter-radio interference from two co-located radios in on channel 1 and 11 in the 2.4 GHz spectrum

NIC-1	NIC-2	NIC-1 Goodput	NIC-2 Goodput	% of Max Goodput
send	silent	5.52	-	-
recv	silent	5.23	-	-
silent	send	-	5.46	-
silent	recv	-	5.37	-
send	send	2.44	2.77	47.6%
recv	send	2.21	4.02	58.3%
send	recv	4.22	2.42	61.0%
recv	recv	4.02	1.89	55.8%

neously. The performance loss is so severe that the benefits of a multi radio system are completely negated. It is evident from the results that the two radios are unable to send and/or receive simultaneously. The reason for this loss of performance is the physical proximity of the radios combined with the closeness of the operating frequencies. Fuxjager et al [92] refers to this as the near-far problem and suggests that there is a serious mismatch in some prior channel selection mechanisms that neglected to include a solution to this problem.

One solution, which is used in [64], is to mount external antennas apart. To implement this, the radios must be capable of accepting external antennas. A problem is that long pigtails might increase attenuation and external antennas may raise costs. Table 3.3 replicates the results of a second experiment by Raniwala et al [64] showing the overall throughput when using external antennas. It is demonstrative of the effectiveness of external antennas to reduce inter-radio interference.

Table 3.3: Inter-radio interference from two co-located radios in the 2.4 GHz spectrum with external antennas

NIC-1	NIC-2	NIC-1 Goodput	NIC-2 Goodput	% of Max Goodput
send	silent	5.93	-	-
recv	silent	5.75	-	-
silent	send	-	5.96	-
silent	recv	-	5.78	-
send	send	5.52	5.96	96.6%
recv	send	5.37	5.89	96.2%
send	recv	5.42	5.41	92.5%
recv	recv	5.66	5.17	93.9%

My solution is to utilize frequency effectively by separating co-located radios by as much spectrum as possible. Table 3.1 shows that there are 3 bands allowed for unlicensed use with 802.11. By using one radio in each of the ISM, UNI-1 and UNI-2 bands, inter-radio interference can be eliminated without the deployment problems of external antennas. Large channel separation enables nearby radios to transmit or receive concurrently. However, based on these conclusions the architecture is limited to a maximum of 3 radios. In order to use more radios, additional frequency must be provisioned, antennas must be mounted apart, or highly directional antennas must be used. In the long term, the ratification of 802.11y may allow additional frequencies for ISPs.

3.5.3 Flexible architecture

This architecture has been designed with neighborhood networks, city

wide wireless deployments, disaster recovery and developing world communications in mind. The general assumption is that there will be one or more gateways to the Internet. Multi radio nodes running OLSR and Routing Driven Channel Selection (RDCS) will form the backbone network, relaying traffic across multiple hops to the gateway.

3.6 RDCS

3.6.1 Introduction

The aim of channel selection in multi radio mesh networks is to create a channel assignment that will provide the highest available throughput while simultaneously ensuring a completely connected mesh. This section will further explore this aim; investigating the delicate balance of connectivity, contention and data rates. In this chapter, a new channel selection mechanism, known as Routing Driven Channel Selection (RDCS) is created and tested.

The first requirement, connectivity; is the connectedness of the mesh network. A network is connected if every node has a working path to every other node in the mesh. For example, in Figure 3.8, the network has become disconnected. The disconnected nodes are now part of a radio island; unable to communicate with the rest of the network. A channel selection algorithm must recognize that connectivity has been broken and restore connectivity. One approach, which will guarantee connectivity, is to assign every node the same set of channels. While this will ensure a fully connected mesh it will result in large amounts of contention, reducing the potential capacity or total

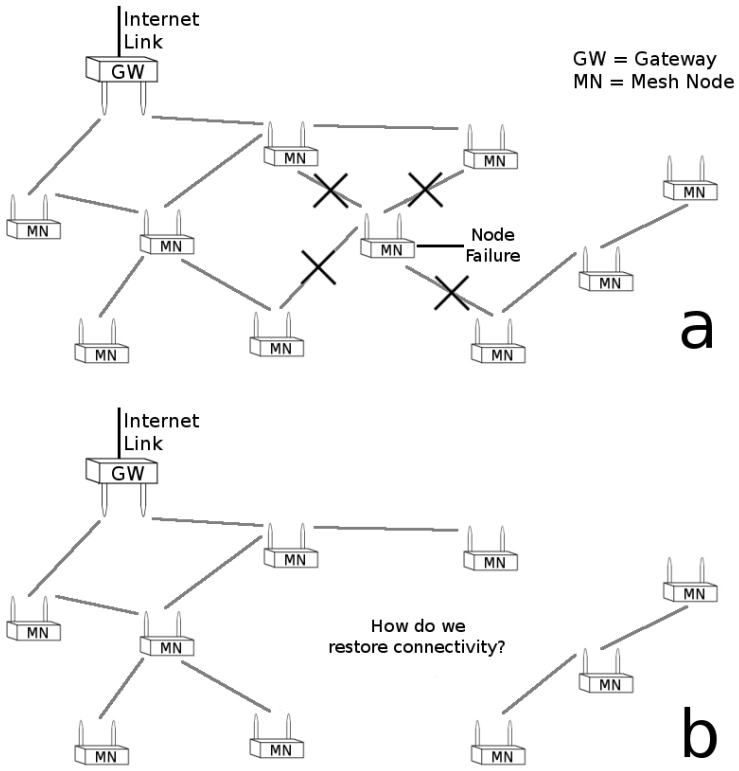


Figure 3.8: Restoring connectivity

throughput.

Contention is the number of nodes competing for use of the medium. As 802.11 is a broadcast technology, access and hence bandwidth is divided between all competing stations. If two wireless nodes are transmitting concurrently on a shared channel, each station will receive approximately half of the links potential bandwidth. The introduction of a third node on the same channel will further reduce the available bandwidth. The solution to this problem is to assign radios to different channels which will reduce contention and improve aggregate bandwidth. A naive channel assignment mechanism might randomly assign channels to radios. However, because two radios must be on the same channel to communicate, a random assignment may not ensure a connected mesh. Hence, this approach would not meet the previous requirement of connectivity.

In addition to the connectivity and contention trade-off, it is equally important for channel assignment algorithms to select high bandwidth links. The data rate of wireless links are determined by a number of factors. Firstly, it is dependent on the type of wireless radio. Older 802.11b wireless radios operate at data rates of 1, 2, 5.5 and 11Mb/s. 802.11a and 802.11g wireless radios operate at rates between 6Mb/s and 54Mb/s and the 802.11n standard offers even higher speeds. The data rate is also largely based on the link quality or SNR (Signal-to-Noise Ratio). The SNR between two nodes is affected by distance, interference and physical obstructions. 802.11 radios automatically adapt the modulation and therefore the data rate to the conditions[7]. Balancing contention, connectivity and data rates is a difficult

[7] An implementation difficulty is that the exact data rate at which a link operates is difficult to know in advance without actually changing to the correct frequency and

task and has been attempted by many studies [89, 90, 86][82, 83, 84, 64, 85] with varying approaches. Our approach to the problem is known as RDCS (Routing Driven Channel Selection).

3.6.2 RDCS implementation

RDCS is a distributed channel selection algorithm written in the Ruby programming language. It is designed to operate with OLSR. RDCS chooses channels intelligently by leveraging information from the wireless driver, the OLSR routing Table and also by exchanging information with neighboring nodes[8].

Many prior channel assignment schemes [82, 83, 84, 64, 85] formulate their problem as a graph problem and optimize for a specific network variable. RDCS differs from these approaches because it is distributed and operates only on locally available information. Furthermore, it does not begin by optimizing for a specific variable, but by canceling out the frequencies which are unable to be used. RDCS then makes channel decisions based on information provided by the driver. The decision for each channel is iterated over with specific mechanisms to provide stabilization/optimization. The implementation discussion begins with an explaination of the inputs and sources of information used by RDCS.

---

transmitting messages. Even with knowledge of the SNR it is inaccurate to correlate the SNR with a specific data rate.

[8] Neighboring nodes: A directly connected node or a 1 hop neighbor

3.6.3 Inputs and RDCS

RDCS has three main sources of information which it uses to make channel decisions. These are: the wireless driver, the routing protocol and directly connected one hop neighbors. These inputs are processed by Ruby regular expressions.

A built in function of 802.11 devices is the ability to scan and find other wireless devices. The scanning process was standardized in the 802.11 specification and has been studied in a number of works [93, 76, 94, 95, 96]. While good scanning implementations are likely to result in better topologies and faster convergence, RDCS can also operate without any scanning functionality. Utilizing the scanning cache is relatively simple, a package known as wireless-tools [97] provides the scanning information from a range of open source drivers allowing RDCS to be driver independent.

Another source of information is the OLSR routing table and OLSR neighbor table. These tables can be probed by external applications through a plug-in provided with OLSR called txtinfo. This plug-in provides a number of informational OLSR tables. RDCS uses the OLSR routing table to discover whether an interface is currently providing unique routes. If the interface is providing routes then, RDCS is likely to stabilize on this frequency/channel. The specific details of this mechanism will be provided in section 3.6.6.

Information can also be retrieved from directly connected neighbors running RDCS. Neighbor's IP addresses can be discovered though the OLSR link table. These addresses are used to setup the exchange of information between neighboring RDCS nodes. The need for this will become clear later

in this chapter. The following section discusses the key features of RDCS including a basic description of these features. My explanation of RDCS begins with one of the first operations it performs; channel masking.

3.6.4 Channel masking

RDCS does not begin by assigning frequencies to radios, but instead, by trimming or masking the list of available frequencies. Channels are deemed inappropriate for a number of reasons including: to avoid rogue nodes, to manage inter-radio interference and to prevent dual links forming between two nodes. Together, these inappropriate frequencies are used as a mask against the list of usable frequencies. This section investigates the reasons for such frequency masks and their implementation. While reading this section the reader should be mindful that a separate channel mask is created for each radio interface.

3.6.4.1 Rogue node avoidance mask

The rogue node avoidance mask searches for channels used by nodes that are not currently part of the mesh network. Most wireless drivers implement a function called background scanning which enables currently unused radios to search for other wireless networks. This information is stored in a scan cache by the wireless driver and is retrieved by wireless-tools [97]. RDCS will retrieve this list and search for nodes not using the specified SSID and nodes not operating in ad-hoc mode. These nodes are regarded by RDCS as rogue nodes and their frequencies are added to the rogue node avoidance mask.

RDCS applies this mask to the list of available channels to avoid sharing frequencies with non-mesh nodes. Sharing a channel with APs can result in interference and less channel time to transmit frames, degrading throughput.

3.6.4.2 Inter-radio interference mask

In section 3.5.2, a phenomenon known as inter-radio interference was discussed. It was found that the frequency bleed or inter-radio interference that can occur between radios deployed in the same router is detrimental to performance. The approach to solving this problem is to ensure that radios are adequately separated by frequency. I recommend that radios deployed in the same device are separated by as much frequency as possible.

To understand the inter-radio interference mask, the the configuration of interfaces and frequencies in RDCS.conf will be briefly described. For RDCS to operate, the number of frequency bands configured in RDCS.conf must equal the number of 802.11 interfaces that will be used in RDCS. For example, if there are two interfaces, ath0 and ath1, [5.18, 5.2, 5.22, 5.24, 5.26, 5.28, 5.3, 5.32] could be specified as as the first band and [5.745, 5.765, 5.785, 5.805, 5.825] as the second band. The first interface will be assigned a frequency in the first band and the second interface will be assigned a frequency in the second band. When RDCS creates the inter-radio interference mask, for each interface, RDCS will check the alternate interface to determine the band currently in use. The inter-radio interference mask will subsequently mask the entire band of frequencies used by the alternate/other radio. By banning the list of channels used by the alternate radio, RDCS can prevent the two interfaces from ever sharing the same band; eliminating inter-radio

interference through distinct channel separation.

3.6.4.3 Dual-link mask

The aim of the dual link mask is to prevent two mesh nodes from converging on the same two frequencies. The scenario whereby two nodes acquire the same channel assignment may reduce the globally available bandwidth due to increased contention. Figure 3.9a, shows a situation that could arise without the dual link mask. In Figure 3.9b the problem has been solved by masking the channel of the neighbors alternate channel. To provide this function, each node must check the channels used by its current 1-hop neighbors. This check is performed by extracting 1 hop neighbors from the routing table and using TCP network sockets in Ruby. These channels are added to the dual-link mask. The reason for the dual link mask is that there is little benefit from forming a secondary link with a neighbor. The aim of the dual link mask is to form a mesh with greater channel diversity as shown in Figure 3.9b.

The dual-link mask is implemented by obtaining the address of a nodes 1-hop neighbors through OLSR. Nodes request channel information from these addresses and add the channel of the alternate radio to the channel mask. Once the rogue node, inter-radio interference and dual link channel masks have been obtained, they are applied against the list of available channels. After performing channel masking functions, RDCS must cost the remaining channels to choose the best channel.

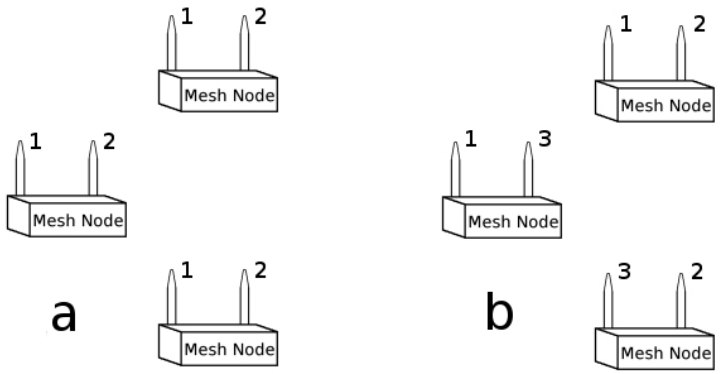


Figure 3.9: Dual-link masks

3.6.5 Channel costing

RDCS costs channels to decide which of the remaining channels should be assigned. Potential wireless neighbors are discovered in the same manner that rogue nodes are discovered. The wireless drivers scan cache is searched to find nodes in ad-hoc mode with the correct SSID. The channels of these nodes are recorded. The remaining usable channels, after the channel mask has been applied, is compared against the list of nodes found in the scan cache. Each channel containing one or more mesh nodes is evaluated based on a simple cost metric.

The RDCS costing scheme provides a rough estimate of which channel will provide the highest throughput for the network. However, the task is inexact because the information gathered by the driver only specifies a SNR and not a link speed[9]. To provide high data rates and low contention, the cost function should give low costs to channels with few nodes and high

[9] This measure/metric could possibly be improved with a sub-study to determine the which SNRs correspond to what data rate. However, the results may not be applicable to all wireless cards. Different cards may have different sensitivities.

SNRs. Equally, it should give high or unfavorable weights to channels with many nodes and low SNRs. The formula shown in equation 3.5, says that the estimated bandwidth of a channel is equal to the average SNR of all nodes sharing a channel, divided by the number of nodes. This formula is not necessarily optimal, however, no attempts were made to further optimise or evaluate the formula because, at the time of experimentation, the SNR reports obtained from the driver were inaccurate.

$$EstBW = \frac{SNRavg}{NoNodes} \tag{3.5}$$

3.6.6 Stabilizing/optimizing channel selection

A problem with RDCS, as it has been discussed thus far, is that it may forever iterate between interfaces, constantly selecting different channels for each interface[10]. Subsequently, a stable channel assignment may never form, causing problems for routing protocols and requiring repeated Dijkstra recalculations. To provide the stability required by the routing protocol, RDCS must have mechanisms to stabilize on channels and prevent constant reassignment. Equally, RDCS must also be capable of recognizing poor channel selections or topology changes and take steps to improve the channel assignment. However, with only local information available, the effect of channel changes on the global network is difficult to predict. The channel stability mechanisms are closely related to a problem referred to in prior work as

[10] Highly likely given the inaccuracy of background scanning mechanisms.

cascading channel changes [91] or the ripple effect [98].

The ripple effect is the potential knock on effect that occurs when a channel change causes a cascade or ripple of channel reassignments throughout the network. Some schemes [89] circumvent this problem by allowing a finite number of channel changes. Similarly, centralized approaches address the ripple effect by terminating the algorithm after an approximate channel assignment has been found. The solutions which terminate channel assignments to provide stability are inadequate as they prevent the network from responding to future network/topology changes. A more adaptable solution is provided by the gateway centered approach; Hyacinth [90]. In this approach, upstream nodes set the channel. Topology changes or breaks enroute to the gateway, cause downstream nodes to search for and connect to another upstream node offering the new lowest cost path to the gateway. This approach is ideal because it stabilizes its channel assignment yet also retains the capacity to respond to future topology changes. A significant problem for many approaches is that the individual nodes will have asymmetric views of the same link. RDCS circumvents this problem by using the symmetric routing protocol metric and subsequently, it will be valued equally by both nodes responsible for the link.

Preventing the ripple effect is similar to the RDCS goal of providing a mechanism to reliably stabilize or optimize channel assignments. Assignment schemes should stabilize good channel assignments whilst simultaneously optimizing poor channel assignments with minimal impact to the surrounding network. Stabilizing or optimizing channel assignments must be balanced, and consequently, any solution will be a trade-off.

In RDCS, channel assignment is evaluated on a per interface basis. An

interfaces channel assignment is regarded by RDCS as stable if the interface currently provides a unique route in the routing table. Equally, a channel is reassigned or optimized depending on whether the interface is providing a route in the routing table. Whilst this scheme seems simplistic, its stabilization and optimization of channel assignments is best appreciated with an example.

In Figure 3.10a the first interface, ath0, on node A connects to node C through the trees with a metric[11] of 8. This route will be stored in the routing table and consequently is considered by RDCS as a stable channel assignment. During this time, node B finds a line of sight link to node C with a metric of 1 (see Figure 3.10b), and, when the second interface on node A is evaluated, it connects to node B with a metric of 1 (see Figure 3.10c). As node B now has a route to C with a metric of 1, node A will learn of this new route. As the path ABC has a metric of 2 and AC has a metric of 8, the new lower cost route will be installed in the routing table and the AC route on ath0 will be removed from the routing Table. Because RDCS only considers interfaces with routes in the routing table as stable, the link between node A and node C will be reassigned; perhaps forming a link with node D (see Figure 3.10d). If node A uses one interface for a route to node D and one interface for a route to node B they will both be considered stable.

The example in Figure 3.10 shows how RDCS optimizes channel selection based on the information provided by the routing protocol. In addition to optimizing routes, this behavior will also adapt to physical changes in the environment. For example, given the previously described example, the

[11] Keep in mind that RDCS does not have a metric of its own. It will optimize for the metric of the routing protocol

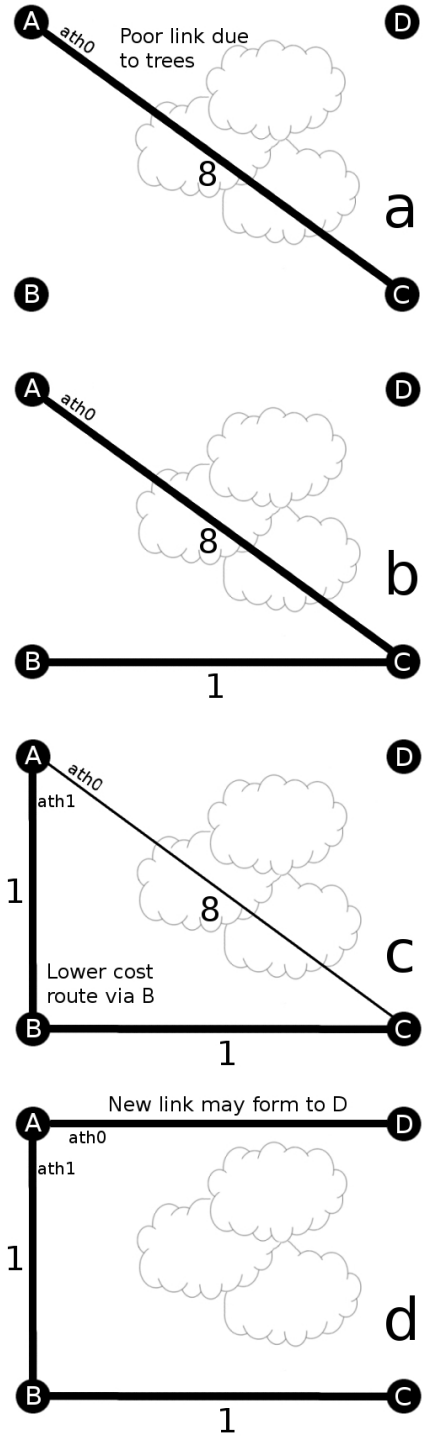


Figure 3.10: Example topology formation

choice AC could initially have been a low cost link, with a metric of 1. However, after some time, if the link conditions deteriorated, the interface would undergo reassignment/optimization. A major advantage of using the routing protocol to decide whether a channel assignment should be kept is that RDCS does not need a metric. RDCS will optimize for the metric used by the routing protocol.

The above example shows *how* RDCS uses the routing table to either stabilize or optimize a channel assignment. This section will explain *why* basing channel assignments on the OLSR routing protocol works. OLSR is a link state routing protocol and subsequently, routing decisions are based on the Dijkstra algorithm. When Dijkstra runs, each node computes its own Shortest Path First tree to every other node node. The tree formed by each node is installed in the routing table. Figure 3.11 shows a sample topology. Each node in this topology computes its Shortest Path Tree (SPT) to every other node in the network. The SPTs for nodes A, B, C and D are shown in Figure 3.12a, 3.12b, 3.12c and 3.12d respectively.

Although each nodes SPF tree is different, on a local scale, Dijkstra ensures that all nodes unanimously agree on the value of links for which they are responsible. In Figure 3.12a and Figure 3.12c, notice that node A and node C are both in unanimous agreement that the link AC should be included in the routing table. It is irrelevant what other nodes believe about the importance of this link because the nodes which *are* responsible for the link, both install it in the routing table. Equally when comparing the Dijkstra trees formed in Figure 3.12a and Figure 3.12b, node A and B both put the link AB in their routing table. Dijkstra ensures that two nodes will always agree on the shortest path between one another. For a final example, Node

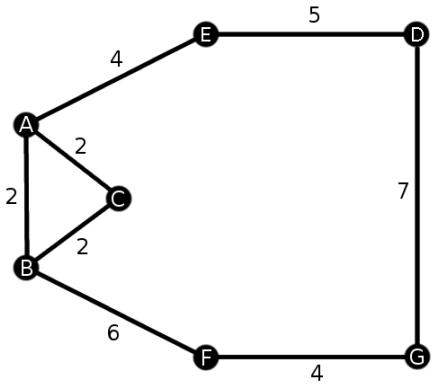


Figure 3.11: Example topology

D is in agreement with nodes A, B and C over the shortest path between itself and the other nodes. With Dijkstra, every node in the network will always agree on the best route between one another. Subsequently, there will always be unanimous agreement, between the nodes responsible for the link, over whether a link should be kept as a stable link used for routing, or, whether it should be reassigned. This mechanism means that RDCS channel re-assignments will be agreed upon by all nodes which helps to prevent a cascade of changes throughout the network.

3.7 Connectedness and Channel Diversity

This section discusses the connectedness and channel diversity trade-off. The greater the channel diversity, the lower the contention and the more simultaneous transmissions can occur. The Hyacinth approach [90] attains maximal channel diversity by allowing both interfaces to be switchable, however, the connectedness constraint is significantly less demanding than other approaches. Channel diversity is attained in Hyacinth because each nodes

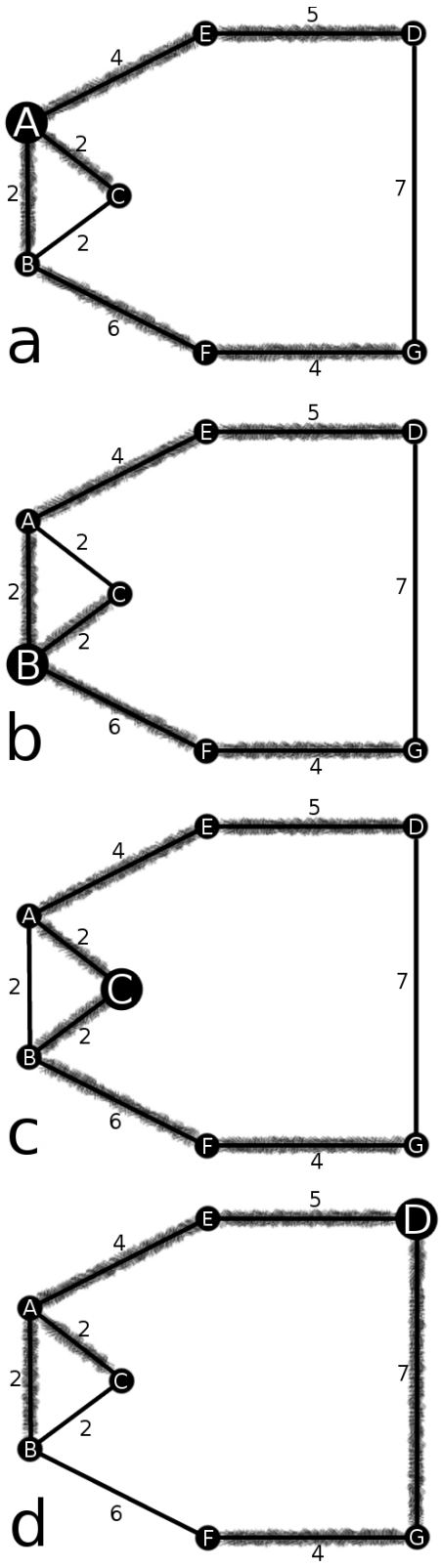


Figure 3.12: Dijkstra shortest paths from the perspective of different nodes

up-NIC will switch to the channel offering the greatest capacity to the gateway. As Hyacinth forms independent spanning trees, multiple gateways will result in the formation of multiple trees. These trees are not connected via the mesh and hence do not provide connectedness, however, it is also arguable that communication can occur via the wired network.

DCA [89], operates by using one radio on a common control channel while the other radio is dynamically switched in attempt to globally reduce the level of interference. As a result of the common control channel, DCA will be unable to attain the same levels of channel diversity as schemes such as RDCS or Hyacinth where both radios are switchable. The advantage of DCA is that mesh connectedness is guaranteed. Furthermore, DCA is routing protocol independent, which provides a significant advantage over both Hyacinth and RDCS.

RDCS ensures connectedness by demanding connectivity to a particular user defined gateway node. In many cases, if the network is simply designed to provide Internet connectivity, the IP address of an Internet server such as www.google.com can be specified. However, if a big distributed neighborhood network is desired, then users can choose a single node to act as the gateway. Subsequently, all nodes will require connectivity to this node before considering their current channel assignment stable. If every node can reach one common node then at least one route will exist between all nodes. RDCS's use of a common gateway operates independently from the routing protocol. The use of a single gateway node does not send all traffic via the gateway; it is simply another attribute that is required before a nodes channel assignment is considered stable. Connectivity to the gateway is determined through periodic pings. Nodes which consecutively fail the

gateway ping consider themselves part of a radio island and will randomize their channel assignment. These mechanisms enable RDCS to build a fully connected mesh network without the use of a control channel.

In addition, to utilizing two switchable interfaces, RDCS provides a number of other mechanisms to allow maximal channel diversity. The dual link mask prevents connected nodes from sharing the same two channels. Also, to further provide channel diversity RDCS allows users to specify the number of neighbors that can share the same channel. By default, the channel diversity variable is three; which is an appropriate variable for a network with dual radio 802.11a/b/g cards. This will allow three mesh nodes to share the same channel. Mesh networks with predominantly 2.4GHz radios may need a more conservative/higher channel diversity variable.

3.7.1 Algorithm overview

RDCS operates in an iterative manner following the pseudocode shown in algorithm 3.1. The first operation that RDCS performs is a check of the OLSR routing table to determine whether an interface is providing routes. If the interface is not providing a unique route in the routing table then it will undergo the process of masking channels (see Section 3.6.4), costing channels (see Section 3.6.5) and then assigning the channel to the interface.

If the routing table check determines that a given interface is providing a route to a unique destination then two additional checks are made. Firstly, the gateway is pinged to ensure connectivity and secondly, the node will check its neighbors to ensure that the number of nodes sharing its channel are below the channel diversity threshold. Upon failure of these checks,

Algorithm 3.1 RDCS algorithm

```
Is the interface in the routing table?

    No

        mask channels

        cost channels

        assign channel

    Yes

        gateway access?

            Yes -> do nothing

            No -> randomize channel assignment

        channel diversity?

            Yes -> do nothing

            No -> mask, cost and assign...
```

RDCS will reassign/randomize the interface by putting the interface through the channel selection process of masking channels, costing channels and then selection. RDCS code can be found at *http://www.brigdingthelayers.org/*. When looking through this code, be mindful that some of the functions that enable OLSR and RDCS to operate smoothly make the code more complex than the pseudocode may infer.

3.8 Implementation Issues & Discussion

Before discussing the results, the implementation issues and drawbacks of RDCS will be jointly discussed. Many of these issues can also be considered as future work. One factor that impedes performance is OLSR's metric. Currently, OLSR uses ETX which is a reliability metric. While ETX is capable of picking the more reliable of two links, it is unable to incorporate bandwidth. Section 3.6.6 provided an example of how a link with a very poor metric would be removed from the routing table and be given a new channel assignment. This functionality enables a poor channel selection to be reassigned or optimized to a new and possibly better channel. Using the current metric, this function cannot function because OLSR will never prefer a high quality 2 hop path over a poor single hop path. When the effects of bandwidth can be incorporated into the OLSR metric, improvements in RDCS should also be seen.

Another problem is the inaccuracy of MadWiFi scanning mechanisms. As this information is used to find neighboring nodes, the time required for network convergence may increase. Furthermore, suboptimal channel assignments can result. Coupled with the previously mentioned metric issues, where OLSR does not represent the bandwidth of the link, poor channel assignments may be stabilized on, forming inefficient topologies. As scanning in wireless drivers improves, RDCS will converge faster and perform better.

In addition to these implementation issues, there is currently no working layer 2 QoS mechanism for ad hoc mode wireless interfaces. With network congestion, OLSR hello messages can be delayed causing the ETX metric to rise and occasionally lead to route changes. There are also a number of un-

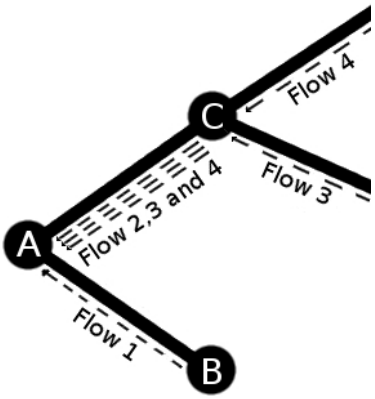


Figure 3.13: Flow unfairness

resolved transport layer issues. Firstly, flow based unfairness is caused when two or more nodes with disproportionate numbers of flows are competing for media access. In Figure 3.13 there are two nodes, B and C which are directly connected to the gateway. Ideally, the flows 1, 2, 3 and 4 should share the bandwidth equally. However, if node C and Node B are both sharing the same channel, the 802.11 MAC layer will provide contention resolution between the two nodes. Subsequently, flow 1 will consume half of the channel time leaving the other half to be shared between flow 2, 3 and 4. These issues can result in unfair bandwidth allocation.

Another problem, which may exacerbate the previous issue of flow based unfairness is RTT unfairness. TCP congestion windows are highly dependent on RTT (Round Trip Times). Flows originating close to the gateway will have significantly lower RTTs than flows originating further from the gateway. While transmission and propagation delays in 802.11 networks are generally low; contention, congestion and a multi-hop topology can result in highly varied RTTs. Flows originating near the gateway with lower RTTs will acknowledge transmissions faster hence building the congestion window

faster. Nodes originating further away from the gateway will build their window more slowly and initially, will get a lower amount of bandwidth.

A similar and related problem to RTT unfairness is link quality unfairness. Frames transmitted by nodes further from the destination must cross multiple links. In addition to increasing the RTT, the drop probability is also increased. Transmissions that must traverse four hops are more likely to be dropped than transmissions that traverse only one hop. While 802.11 uses acknowledgments to ensure reliability, it does not prevent TCP timeouts in the presence of transmissions that must traverse multiple hops.

As a result of these issues, an experimental conclusion was that the nodes directly connected to the gateway quickly capitalized on the available bandwidth starving more distant nodes of bandwidth. To improve the consistency and fairness of the results, the Internet gateway rate limited each node to allow 5Mb/s and bursting bursting up to 10Mb/s. Such rate limiting techniques are congruent with the methods used in many community mesh networks.

3.9 Testbed and Results

The testing platform consisted of 8 ALIX wireless nodes, each with two 802.11a/b/g network cards. The ALIX platform, is an embedded x86 PC with dual miniPCI Slots. The operating system used was Voyage Linux and the wireless cards were Atheros CM9s running a modified version of MadWiFi v9.3.3. OLSR version 5.6 was used with the txtinfo plug-in installed. Two different tests were performed to mimic the potential traffic types of such networks. The first test was based on an Internet access traffic model

where all traffic is directed towards a gateway. In this test, a wired server that was connected to the wireless gateway, performed simultaneous iperf tests to every wireless node. Ten different topologies, both dense and sparse, were tested. The environment was a set of university offices. Nodes spanned between two different floors. The ten different topologies were all random placements of nodes in offices. Both dense and sparse topologies were tested. RDCS was compared with single radio networks, where all nodes are statically assigned to the same channel, and also multi radio networks, where every node is given two channel assignments. Each individual test and scenario was performed 5 times and the results were averaged.

A cursory glance at Figure 3.14 reveals variations in the performance of both the multi radio and RDCS scenarios. While bandwidth variations between topologies were anticipated, the extent of variations between the performance of single radio, multi radio and RDCS scenarios were unexpected. Some of the variations are a result of the routing protocol being channel unaware. Given a multi channel network configuration, OLSR may sometimes prefer a single channel throughout the network resulting in little difference between the single channel and multi channel configurations. In other cases, it utilized and balanced the multiple channels resulting in significantly higher throughput for the multi radio configuration. Because RDCS forces a channel diverse assignment, it is less susceptible to these issues, however, metric limitations still impeded certain RDCS functions. RDCS's inability to differentiate between links with disproportionate bandwidths in some cases led to stabilization on channels that were reliable yet can only provide a few mega bits per second of actual throughput. This occasionally lead to poorly

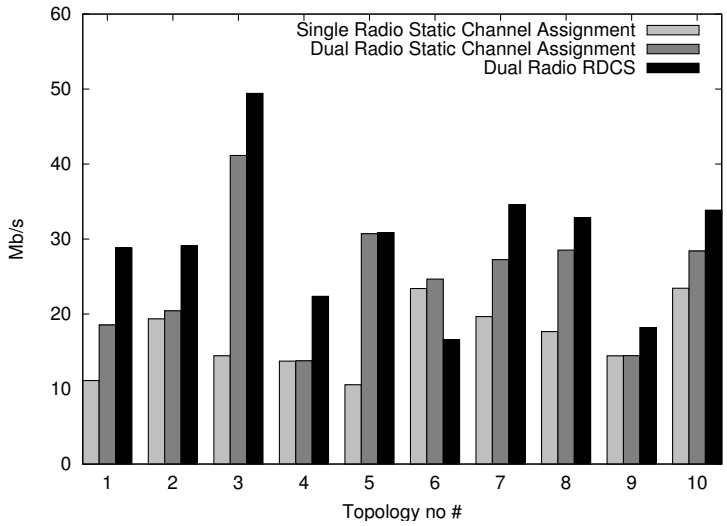


Figure 3.14: Bandwidth test to gateway

performing topologies.

The second test was performed from the perspective of a community network; perhaps where the dominant traffic model is a file sharing or Peer-to-Peer (P2P) traffic model. Considering the volume of peer-to-peer traffic in modern networks, these scenarios cannot be ignored. To test random traffic patterns, a Ruby program was built whereby each node randomly starts iperf sessions with other nodes. A new iperf session was started at a random time between 5 and 15 seconds with each iperf session lasting 10 seconds. This program looped 30 times and the bandwidth recorded by each random iperf session was averaged. As these tests were performed over a much longer period of time and are less dependent on the luck of links formed to the gateway, the results are more systematic: even over a range of topologies. Similar to the the previous results, RDCS outperforms both single radio and multi radio scenarios.

Although the results are encouraging, with a larger testbed, the difference

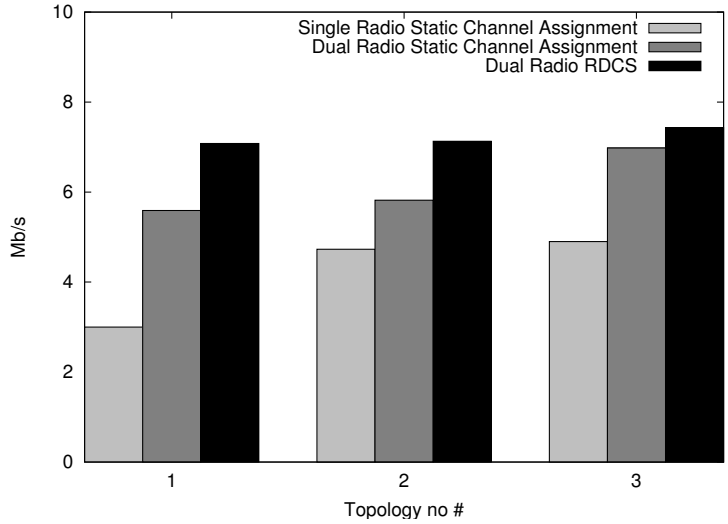


Figure 3.15: Random P2P bandwidth test

in these results would be more pronounced. With static channel assignments, larger numbers of nodes will increase contention and reduce the bandwidth available to each individual node. Mechanisms such as RDCS, that produce channel diverse topologies, will suffer from comparatively less contention as the number of nodes increase. Larger mesh networks will exacerbate the performance difference between multi-radio assignments and static channel assignments.

Additionally, when OLSR adopts an ETT metric that incorporates the effects of bandwidth, RDCS will be able to make more intelligent optimizations regarding the channel assignment. In conclusion, while RDCS can currently outperform static channel assignments, in the future, the difference between the two schemes will grow.

An important criticism of RDCS, and perhaps of other channel assignment schemes, is that by aggressively minimizing interference with a scheme that only allows a certain number of nodes to share a given frequency, the

number of links being traversed may be artificially increased. Channel assignment schemes such as RDCS may force multi hopping when two nodes, deployed on the same channel, would be close enough to communicate directly. Despite this potential inefficiency, RDCS was thoroughly tested in dense topologies and even in these scenarios, RDCS outperformed static channel assignments.

3.10 Conclusion

This chapter provides a indepth literature review. With the benefit of hindsight, a new channel selection scheme that attempts to overcome a fundamental performance limitation in 802.11 wireless mesh networks is developed. Theoretical work has shown that in single radio networks, considering a random traffic pattern and placement of nodes, as the number of nodes increase, the bandwidth available to individual nodes approaches zero. The reason for this sharp degradation in bandwidth in a multi hop environment is a caused by a combination of problems including half-duplex transmission, contention, and 802.11's large carrier sensing range. Given that 802.11's economies of scale are responsible for its sophistication and affordability, this chapter has argued that the contention problem could be better addressed by using multiple 802.11 radios embedded within one device.

In this chapter, a new channel selection mechanism, known as RDCS is created and tested. Channel selection in multi radio ad hoc networks is non-trivial because the competing variables of connectedness and interference must be balanced. Furthermore, channel assignments must be performed in a distributed manner based only on locally available information. The

performance of RDCS confirms the benefits of a channel diverse topology, however, before multi radio mesh networks can realize their full potential, many issues remain at all layers of the OSI networking model. 802.11e QoS mechanisms are required in ad hoc mode to prioritize critical routing traffic such as OLSR hellos. Routing protocols must incorporate bandwidth from the data link layer and perhaps eventually channel information into the routing metric. Transport layer unfairness including flow based unfairness and RTT unfairness also require attention. RDCS is capable of increasing spatial diversity by creating multi channel topologies, however, much future work is required to fully utilize the additional capacity.

Chapter IV

802.11 Acknowledgment Efficiency and Solutions

4.1 Introduction

TCP was never designed for lossy wireless links or ad hoc networks, and subsequently, every lost packet is interpreted as congestion. To prevent TCP from invoking congestion avoidance, and thus reducing the throughput of a connection, the link layer must be reliable. 802.11 acknowledgments provide link layer reliability, however, to provide this function they introduce a large overhead. This study investigates alternate mechanisms to hide or recover end-to-end losses.

This chapter uses a theoretical calculation to show the extent of overheads imposed by both 802.11 and TCP acknowledgments. The potential performance gains of removing this overhead are then proven through modifications to the MadWiFi driver. Before experimenting with these mechanisms, the IEEE's solution to 802.11 ack inefficiencies, known as BlockAck is outlined. Numerous other solutions such as congestion control and PEPs (Performance Enhancing Proxies) are also explored. As these technologies are inefficient, or inapplicable for multi hop ad hoc networks, a new cross layer proxy known as D-Proxy is developed. This chapter only considers reliability solutions for TCP, the solution that is eventually proposed could also operate in conjunc-

tion with UDP or other transport layer mechanisms if required.

4.1.1 802.11 overheads

To justify this work, it is important to show the extent of inefficiencies imposed by current 802.11 ack mechanisms. A theoretical model is built to demonstrate the overhead imposed. These results are latter reinforced with experiments. The theoretical model is based on 802.11a/g, however, the results are equally applicable to 802.11b and, increasingly relevant for the upcoming 802.11n. Table 4.1 shows the rates of 802.11a/g modulation. Note that peak modulation in 802.11a/g transmits 216 data bits per symbol. Each symbol in 802.11a/g takes 4μs to be transmitted. This means that 802.11a/g can transmit 250,000 symbols each second. Each one of these 250,000 symbols represents 216 data bits which is 54,000,000[1] bits per second or roughly 54Mb/s.

Unfortunately, achievable throughputs are roughly half this number; approximately 27Mb/s per second. If 802.11a/g has a data rate of 54Mb/s, why are real world throughputs roughly half? Put simply, a combination of MAC layer contention delays, packet headers and TCP acknowledgments. In the following sections, the extent of these overheads will be explained and calculated.

Table 4.1: OFDM transmission characteristics

Tx	Modulation	Coding rate	Coded BPC	Coded BPS	Data BPS
6	BPSK	$\frac{1}{2}$	1	48	24
9	BPSK	$\frac{3}{4}$	1	48	36
12	QPSK	$\frac{1}{2}$	2	96	48
18	QPSK	$\frac{3}{4}$	2	96	72
24	16-QAM	$\frac{1}{2}$	4	192	96
36	16-QAM	$\frac{3}{4}$	4	192	144
48	64-QAM	$\frac{2}{3}$	6	288	192
54	64-QAM	$\frac{3}{4}$	6	288	216

4.1.2 Frame size

Figure 4.1 shows the end-to-end frame transmission requirements for every TCP transaction over 802.11. In wired networks, transmissions are reliable and packet loss usually only occurs as a result of congestion. However, 802.11 is inherently unreliable, and consequently, every TCP segment or packet is acknowledged by an 802.11 acknowledgment. These MAC layer acknowledgments[2] provide a reliable 'Ethernet like' MAC layer and prevent adverse reactions from TCP. Figure 4.1 shows that both the TCP data segment and TCP ack are individually acknowledged by MAC layer acknowledgments. In this chapter, the reliability replication that occurs between TCP acks and 802.11 acks is the starting point of this efficiency investiga-

[1] $216 \times 250,000$

[2] We use the terms MAC layer acknowledgment, link layer acknowledgment and 802.11 acknowledgment interchangeably

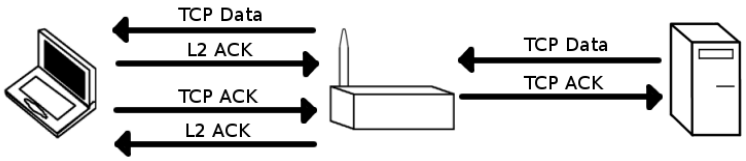


Figure 4.1: TCP over 802.11

tion, however, further discussion will be delayed until the overhead imposed by these functions has been demonstrated. This section aims to familiarize the reader with the 802.11 transmission requirements for an end-to-end TCP transaction.

Figure 4.2 shows the constituent parts of a TCP transaction; the TCP data frame, the TCP ack frame and the 802.11 ack frame. It is assumed that the payloads are broken into 1442 byte segments. On top of this payload is the: 32 byte TCP header, 20 byte IP header and 24 byte MAC layer header which are all incrementally added as transmissions move down the ISO networking stack. Figure 4.2 shows the size of different frames in bytes and more importantly time. The calculations used to determine transmission time are shown in Table 4.2. All calculations measuring efficiency use time in microseconds as it is more representative of overheads than bytes. For medium access functions or physical layer preambles, data and therefore bytes are irrelevant. Furthermore, TCP data frames and TCP ack frames are transmitted at the peak data rate, whereas 802.11 acks are transmitted at lower rates of 24Mb/s or 6Mb/s. It is important to understand that the finite resource in a shared medium is time, and subsequently, all calculations are based on time.

The rationale behind transmitting 802.11 acks at lower speeds is that 802.11 acks are small, yet their loss requires the retransmission of an en-

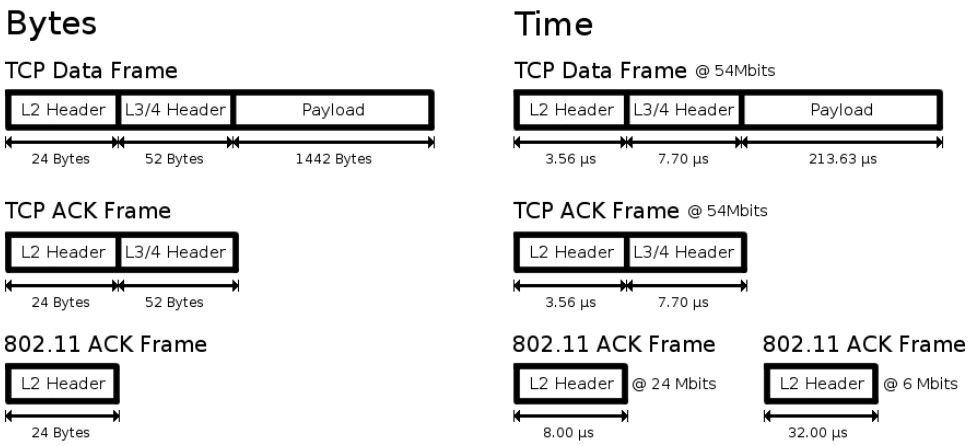


Figure 4.2: Different transmission sizes

tire TCP segment. While acknowledgments transmitted at 54Mb/s might be more efficient than acks transmitted at 24Mb/s or 6Mb/s, some of the overheads such as preambles and Short Inter-Frame Spacing (SIFS) are fixed and therefore doubling the data rate does not equate to the channel time being halved. Perhaps more importantly, the increased reliability from lower bit rate acknowledgments are considered an acceptable trade-off for the marginal loss in efficiency. Now that the frame transmission components have been understood and calculated in Figure 4.2, fixed 802.11 Distributed Coordination Function (DCF) and Physical Layer (PHY) overheads will be investigated.

4.1.3 802.11 DCF and PHY overheads

Overheads in 802.11 networks are more complex than simply the transmissions and header sizes. To complete the TCP data transmission model, the MAC layer, or more specifically 802.11 DCF, delays must be incorporated. To provide contention resolution, 802.11 devices implement random backoff

Table 4.2: Transmission efficiency I

Trans Type	Bytes	Bits	Symbols Required	Time (μs)
Payload @54	1442	11536	53.41	213.63
L3/L4 Header @54	52	416	1.93	7.70
L2 Header @54	24	192	0.89	3.56
L2 Header @24	24	192	2.00	8.00
L2 Header @6	24	192	8.00	32

before transmitting a frame. DCF operates like a low layer QoS and fairness mechanism. It ensures that important management frames can access the medium before data frames and also that access is reasonably fairly shared between multiple stations. After a frame has been transmitted, stations must back off. Important frames, such as beacons and layer 2 acknowledgments, have a short back-off time known as Short Inter-Frame Spacing (SIFS). SIFS ensures that they are sent before less important data frames which use DCF Inter-Frame Spacing (DIFS).

In 802.11a SIFS lasts 16μs [99]. DIFS is always 2 x slot time + SIFS which means that DIFS is 34μs[3] for 802.11a. In addition to contention resolution, 802.11a also includes a 20μs preamble to synchronize the radios and 4μs for the PHY Layer Convergence Procedure (PLCP) which includes the PHY header and trailer. Knowledge of the timing required for MAC layer contention resolution and PHY headers can be added to the combine this with the previously discussed overheads to derive the theoretical throughput of 802.11a/g.

[3] $2 * 9 + 16 = 34$

4.1.4 Total delays

As previously discussed, a single data transmission in 802.11 consists of two data frames, each acknowledged by a layer two ack; and a TCP ack, acknowledged by another layer two ack. These transmissions, and the delays imposed, are shown in Table 4.2. Mindful that layer 2 acknowledgments can be sent at different data rates, this chapter has shown the overheads of TCP transactions with both 24Mb/s and 6Mb/s acknowledgments.

Table 4.3 and Table 4.4, show the total transmission time in μs (microseconds) for the various frames. Given knowledge of packet transmission times, the total throughput can be calculated. The results show that standards based 802.11 networks have throughputs between 27.11×10^6b/s and 29.64×10^6b/s, varying with the data rate used for Layer 2 acks. Note there are two TCP data segments for every TCP ack which is the case for all modern TCP implementations.

4.1.5 Overhead imposed

The introduction alluded that this chapter would investigate the functionality replication in 802.11 and TCP acknowledgments. Therefore, it is of interest that, following the previous calculations, between 18% and 25%[4] of the transmission time is consumed by layer 2 acks. Equally, it is also possible to calculate that the overhead imposed by TCP acks is between 15% and 17%[5]. In the calculation of overheads induced by TCP acks, a layer 2

[4] $\frac{(48+48+48)}{(282.89+48+282.89+48+69.26+48)} = 18\%$ or $\frac{(72+72+72)}{(282.89+72+282.89+72+69.26+72)} = 25\%$

[5] $\frac{(69.26+48)}{(282.89+48+282.89+48+69.26+48)} = 15\%$ or $\frac{(69.26+72)}{(282.89+72+282.89+72+69.26+72)} = 17\%$

Table 4.3: Transmission efficiency with MAC layer acks @ 6 Mbit

Frame @Rate	IFS	Preamb	L2 Head	L3/4 Head	Payload	Total
TCP Data @54	34	24	3.56	7.7	213.63	282.89
L2 ack @6	16	24	32	0	0	72
TCP Data @54	34	24	3.56	7.7	213.63	282.89
L2 ack @6	16	24	32	0	0	72
TCP ack @54	34	24	3.56	7.7	0	69.26
L2 ack @6	16	24	32	0	0	72

Total Transaction	Trans per sec	Trans payload	Throughput
851.04μs	1175.04 packets	23072 bits	27.11×10^6b/s

All times given in microseconds (μs)

Table 4.4: Transmission efficiency with MAC layer acks @ 24 Mbit

Frame @Rate	IFS	Preamb	L2 Head	L3/4 Head	Payload	Total
TCP Data @54	34	24	3.56	7.7	213.63	282.89
L2 ack @24	16	24	8	0	0	48
TCP Data @54	34	24	3.56	7.7	213.63	282.89
L2 ack @24	16	24	8	0	0	48
TCP ack @54	34	24	3.56	7.7	0	69.26
L2 ack @24	16	24	8	0	0	48

Total Transaction	Trans per sec	Trans payload	Throughput
779.04 μs	1283.64 packets	23072 bits	29.64×10^6b/s

All times given in microseconds (μs)

ack was included as part of the TCP ack overhead. Because this chapter is investigating overhead reduction, removing the TCP ack will also remove its associated MAC layer ack.

While these overheads are significant, real world payloads may be smaller than 1448 bytes which will inflate these figures. Smaller packet sizes increase overheads because the greater number of frames transmitted the more MAC layer acks, Inter-Frame Spacing (IFS) and preambles must also be performed. For the same reason, in the future, higher data rates will also increase the relative overhead of these fixed delays. This occurs because, as packets are transmitted faster, the fixed delay components will be performed more often. For these reasons, the suggested overheads may be considered a best case scenario. Given the size of these overheads, removing either TCP or layer 2 acks could result in substantial speed increases.

4.1.6 Similar prior work

MAC layer acks are designed to provide reliability to the unreliable 802.11 link and TCP acks provide end-to-end reliability and flow control. Thus the reliability function occurs at both the data link layer and the transport layer, however, this research is not the first to recognize this duplication.

Pang et al [100] proposed a mechanism to remove the TCP ack. The approach uses the MAC layer acknowledgment as confirmation of the TCP data frame. When an AP receives this MAC layer confirmation, a TCP acknowledgment is generated on behalf of the wireless client. Transmission efficiency increases because TCP acks are no longer generated by the client and no longer traverse the wireless link, however, there are a number of

drawbacks.

Firstly, there is no obvious way to incorporate this into multi-hop ad-hoc networks. This scheme generates a TCP ack as soon as the 802.11 ack is received but, this makes the assumption that there is only one wireless hop. In multi hop networks, it would be incorrect to assume that the sender of the MAC layer ack is the TCP receiver. This scheme also requires changes to both the host and AP. MAC layer acks represent a larger overhead. Furthermore, removing 802.11 acks does not require changes to end hosts, break TCP's end-to-end semantics or 802.11 standards compliance.

Another study by Barcelo et al [101] also questions the benefits of acks in 802.11. This study suggests that 802.11 acks should be removed for VoIP traffic. As VoIP is designed to withstand minor packet loss, they question whether the removal of 802.11 acks could accommodate a greater number of voice calls. Put simply, the efficiency gains of not sending acknowledgments will outweigh any negative effects that packet loss may have on call quality. Their study concluded that in standard 802.11a/g networks, fourteen concurrent calls was the the maximum before congestion reduced call quality. Without link layer acknowledgments, the calls were more efficient, however, when 14 calls were reached, collisions were so high that the Mean Opinion Score (MOS) dropped below 3.5, causing call quality problems. They suggest that an ideal mechanism would switch the VoIP calls to a NoAck policy when the AP experiences congestion.

4.1.7 Standard compliance and removing acks

The previous chapter investigated mechanisms to utilize multiple channels

and was dismissive of the multi channel MAC schemes because they broke interoperability with existing 802.11 DCF. Therefore, it would be hypocritical in this chapter to propose standards breaking modifications like removing layer 2 acks.

Since the introduction of the IEEE 802.11e Quality of Service (QoS) amendment, MAC layer acknowledgments can be removed in a standards compliant manner and therefore, this proposal can operate without breaking standards compliance. The aim of 802.11e is to allow different frames to utilize different IFS, providing QoS service to time sensitive transmissions.

In addition to QoS improvements, it also provides the capacity to specify that certain frames should not be acknowledged. This function was included to prevent highly time sensitive transmissions from being retransmitted and can therefore be used to remove MAC layer acknowledgments in a standards compliant manner. Given that standards compliance has been shown, the next step is to remove 802.11 acks and investigate whether practical results match theoretical estimates.

4.1.8 Real world NoAck performance

Physical experiments were performed to determine the effect of removing MAC layer acknowledgments from 802.11. This was designed to test the accuracy of the previous model and experimentally show that the 802.11 ack overhead is significant. The MadWiFi driver (version 9.3.3) was modified such that the interface would neither transmit or wait for an ack following the reception of a data frame. The experimental topology used is shown in Figure 4.3.

Table 4.5: Link performance of STD 802.11 and NoAck 802.11

STD 802.11	NoAck 802.11
27.4Mb/s	33.5Mb/s

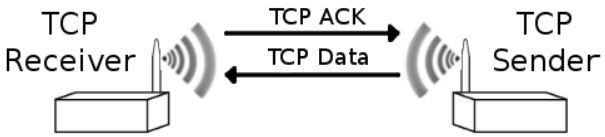


Figure 4.3: Link testing setup

The results, shown in Table 4.5, suggest that unacknowledged 802.11a allows a peak transmission capacity of 33.5Mb/s. Comparatively, the experiment shows that the maximum throughput of standards based 802.11a was 27.4Mb/s. These results suggest that the overhead of 802.11 acks is approximately 22% which is roughly in-line with the previous theoretical calculations which stated that between 18% and 25% of the transmission time is consumed by layer 2 acks.

Unfortunately, these results are somewhat misleading and link layer acknowledgments were implemented in 802.11 for good reason. TCP, which provides flow control and error recovery, interprets packet loss as congestion. Each lost packet results in the congestion window being halved. In addition, the congestion window may not begin regrowing until the lost packet is recovered. Therefore, a single packet loss can have a significant affect on the congestion window of TCP. The testbed, shown in Figure 4.3, was used to obtain the results shown in Table 4.5. This testbed had an end-to-end TCP latency of under a millisecond. As the end to end latency increases, performance sharply degrades. To add latency, a WAN emulator was inserted and

Table 4.6: Link performance of STD 802.11 and NoAck 802.11 with increasing latency

Added Delay	STD 802.11a	802.11a NoAck
0ms	26.54Mb/s	25.52Mb/s
10ms	26.51Mb/s	8.32Mb/s
20ms	26.30Mb/s	6.58Mb/s
40ms	25.28Mb/s	3.94Mb/s
60ms	22.83Mb/s	2.99Mb/s
80ms	19.54Mb/s	1.74Mb/s
100ms	17.12Mb/s	1.65Mb/s
120ms	13.68Mb/s	1.46Mb/s
140ms	13.22Mb/s	1.36Mb/s
160ms	9.04Mb/s	1.17Mb/s
180ms	10.13Mb/s	1.02Mb/s
200ms	8.59Mb/s	1.17Mb/s

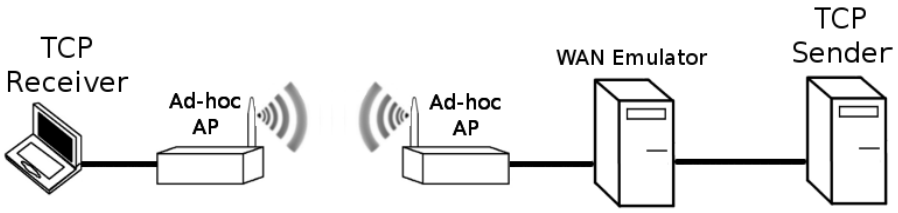


Figure 4.4: Testing setup with added latency

the web server was moved onto a dedicated machine. Identical tests were then performed over a range of latencies. The topology used for the second round of tests is shown in Figure 4.4. The results of this test are shown in Table 4.6.

These results show that as the end-to-end latency increases, the throughput of unacknowledged 802.11 transmissions sharply degrade in comparison

to standard 802.11. Running TCP over a lossy unacknowledged 802.11 is so delay sensitive that that simply moving the TCP endpoints off the APs and onto separate machines and adding a 100 Mbit Ethernet bridge[6] was enough to reduce TCP throughputs from 33.5Mb/s to 25.5Mb/s. Although transmissions without 802.11 acks will be efficient, requiring less channel transmission time to move the data, the data transfers complete more slowly due to under-utilization.

TCP throughputs degrade rapidly with increasing latency because dropped packets are interpreted as congestion, causing the TCP sender to half the congestion window. Large RTTs are the catalyst for two reasons. Firstly, larger RTTs will require more packets to be released unacknowledged from the TCP sender to fill the link. Secondly, paths with larger RTTs, will recover lost packets more slowly. As a result, the congestion window will transmit all of the packets allowed to be outstanding and the missing packet will take longer to be recovered. Lengthy recovery also means that it will be longer until the TCP window can begin regrowing. Based on these results, link layer reliability is a necessary overhead for TCP networks. Perhaps an idyllic solution to this problem is to move from the current 802.11 ack system of positively acknowledging successful packets to negatively acknowledging unsuccessful packets. Such a system would provide reliability and incur minimal overhead.

[6] In this case a WAN Emulator adding 0ms of latency

4.1.9 Positive and negative acknowledgments

Currently, the 802.11 ack system is a positively acknowledging system. For every successful transmission, whether TCP data or TCP ack, a MAC layer acknowledgment is sent. In the standard 802.11 positive ack scheme, nodes use the absence of an ack to indicate a loss. A better approach, if possible, would be to negatively acknowledge packets. Put simply, instead of transmitting acks for every packet correctly received, why not instead, transmit a negative ack for every packet not correctly received.

Currently, this is impossible in 802.11 because the radios cannot send and receive simultaneously. This means that a radio cannot hear if its transmission is being corrupted as it is sent. Furthermore, packet corruption generally happens at the receiver side not the sender side and therefore, even if the sender could send and receive simultaneously, its assessment of transmission success may differ from the receivers transmission success. Figure 4.5 shows the classic hidden node problem where both A and C transmit simultaneously; unaware that the transmission is corrupted at B.

This chapter investigates solutions to the ack efficiency of 802.11 with a specific focus on multi hop ad hoc networks. Solutions from all networking layers are considered. Although there is no obvious way to to turn 802.11 into a negatively acknowledging system at the MAC layer, IEEE standards based work is addressing inefficiencies in the existing ack scheme. These MAC layer efficiency mechanisms are the next subject of inquiry, however, the concept of a negatively acknowledging system will be re-examined as the discussion progresses to higher layer and cross layer solutions to the problem.

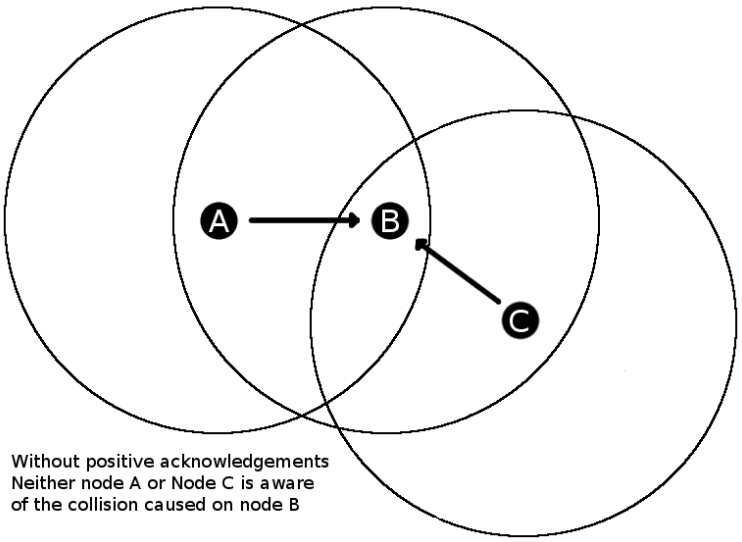


Figure 4.5: 802.11 cannot detect collisions

4.2 MAC Layer Solutions

4.2.1 Future data rates

The material presented thus far is based on the standard 802.11 DCF MAC [3]. This section discusses 802.11e [102] and 802.11n [103] efficiency enhancements that have attempted to address this problem. Using the standard 802.11 DCF, MAC layer inefficiencies will worsen as the data rates increases. This has been recognized for some time [104, 105, 106, 107, 108, 109] and has been addressed in current 802.11e [102] and 802.11n [103] amendments. Ack efficiency is projected to worsen because, as the data rate rises, the fixed overheads such as preambles and IFS will become an increasingly large component of transmission time. Using the previous transmission model, the relationship between achievable throughput and data rate is graphed in Fig-

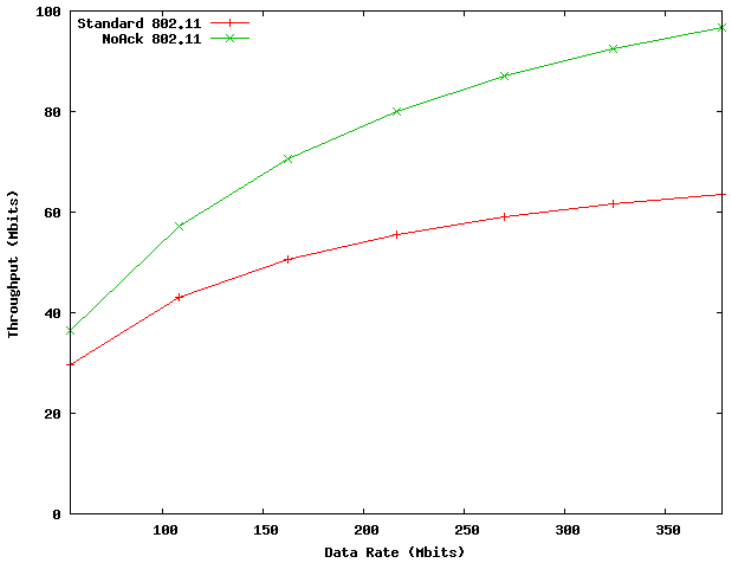


Figure 4.6: Future data rates

ure 4.6. This shows that, using the current 802.11 DCF, increases in data rates do not correspond to linear increases in end-to-end throughput and begin to diminish at higher data rates. Both 802.11e and 802.11n propose amendments to improve efficiency, these improvements are discussed separately.

4.2.2 802.11e BlockAck

The IEEE 802.11e BlockAck function enables senders to transmit multiple packets before requiring an acknowledgment. When packets are acknowledged, a single BlockAck is sent rather than an individual acknowledgment for each packet. The 802.11e BlockAck function can significantly reduce the ack overhead in 802.11.

While there are two types of block acknowledgments, immediate Block-

Ack and delayed BlockAck, for brevity, this chapter only considers the more efficient immediate BlockAck. Using block acknowledgments has a number of benefits. The reduction in the number of acknowledgments transmitted is the most obvious. Figure 4.7 demonstrates the difference between standard 802.11 and BlockAck 802.11. Note that the IFS between the data frames in BlockAck in Figure 4.7 is SIFS rather than DIFS. Subsequently, the use of BlockAck saves on IFS as well as the number of transmitted acks.

The protocol operation is relatively simple, the number of frames that can be sent in a block is defined by the AP during the BlockAck transmission setup. Two new 802.11 frames are required for setup and tear-down, Add BlockAck (ADDBA) and (Delete BlockAck) DELBA. In the case of immediate BlockAck, when the sender has finished transmitting a block of packets, it will send a BlockAck request to the receiver. The receiver will then respond with a BlockAck to acknowledge the successfully received frames. Any frames sent but not acknowledged will be resent as part of the next block of transmissions. If a frame is lost, the receiver will buffer the packets received until the lost frame has been recovered. Buffering and then reordering of packets is done so that packets can be passed to the upper layers in order.

A reduction in acks and IFS delays can significantly reduce overheads, however, similar to the previous calculation, the reduction in efficiency is difficult to precisely identify as it will vary with a number of factors. Some studies suggest that for bulk transfers with large packet sizes, the performance benefit of BlockAck is approximately 10% [105, 104].

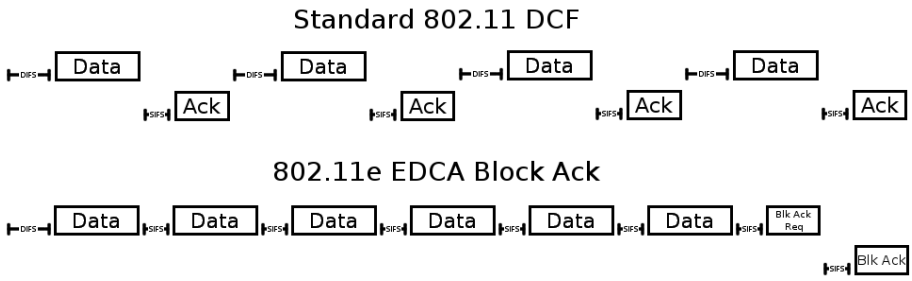


Figure 4.7: Standard 802.11 DCF vs 802.11e BlockAck

4.2.2.1 802.11 BlockAck: delay vs efficiency

Most academic work suggests that BlockAcks increase the throughput but also express concern over the number of packets transmitted in a block because of the affect on time sensitive traffic [104, 105, 106, 107, 108, 109]. Obviously it is more efficient to transmit a larger number of packets in a block because larger blocks mean that fewer layer 2 acks must be transmitted. However, the problem with transmitting larger blocks is delay. There are two types of delays that may be problematic.

The first type of delay is simply the media sharing or contention delay. A old argument for limiting the size of packets transmitted over any medium is that large packets take a longer time to serialize. If a time sensitive packet, such as a voice packet, gets queued behind a large data packet, jitter may become intolerable. BlockAcks allow up to 64k of data to be sent in a block; reviving the old media sharing/contention argument. With this amount of data being transmitted, calculations suggest that 64k of data, not including MAC back-off, preambles or acks, just data, at 54Mb/s will take 9.5ms to serialize. A voice packet that loses contention to other stations transmitting

64k blocks may cause an intolerable amount of jitter. Admittedly, definitions of tolerable delay will differ with different networks and applications; the argument is that larger block sizes, which are most efficient, may not facilitate fair and jitter free networks.

The second disadvantage is the delay imposed when waiting for frame recovery. Frames that are lost within a block will temporarily prevent subsequent frames from being delivered to the upper layers. This buffering function is performed to maintain the transmission order of data. Due to the way the medium is shared, any lost packet(s) may not be retransmitted until the station has re-won contention for the medium. If other stations win contention and transmit large blocks of data, there may be a considerable delay before contention is re-won and the lost packet(s) can be retransmitted. Thus, by allowing block transmissions, the time required to recover lost packets and pass them to the higher layers can considerably increase. It also means that when missing packets are successfully resent, an entire block of packets will be simultaneously passed to the upper layers causing potential TCP compression problems. For these reasons, block size is a trade-off between efficiency and delay [104, 105, 106, 107, 108, 109]. Cabral et al [104] believes that due to the delays added by BlockAcks, the ideal size is between 12 and 16 packets.

A number of other details concerning BlockAcks were also raised in prior work. Li et al [107] suggests that the potential gain from BlockAcks is reduced at lower data rates and larger packet sizes and hence the gains purported by academic studies may not be as large as in real networks where transmission speeds are below 54Mb/s. Ranjitkar et al [109] believes that the introduction of BlockAck in 802.11s mesh networks could worsen performance because of

the increased possibility of hidden nodes. They believe that the lengthy medium reservation of large blocks will lead to under-utilized resources and fairness problems.

A limitation of many prior BlockAck studies [104, 105, 106, 107, 108, 109] is that they have all been either mathematically modeled or simulated. This has provided extensive understanding at the MAC layer and potential performance benefits, however, to my knowledge, no study has yet practically investigated the protocol on a real network and documented end-to-end TCP performance.

4.2.2.2 802.11 BlockAck: unexplored TCP interaction

TCP, the end-to-end protocol that carries the majority of Internet traffic, is responsible for flow control and mediates the transmission of data using a window scheme. MAC layer interaction with TCP is very important as TCP determines the end-to-end transfer rate. TCP has a self clocking mechanism whereby returning TCP acknowledgments prompt the release of new data segments onto the medium. The goal of TCP is to recover errors and facilitate the fastest and most fair transport of data. To do this, the release of TCP data packets onto the medium should be smooth and paced. Excessive traffic bursts are problematic for TCP as they may suddenly increase queue sizes, causing packet drops. A lot of work has gone into reducing the natural traffic bursts caused by TCP [110, 111, 112, 113]. This symptom is often referred to as ack compression [111]. Sundaresan [114] made specific reference to this problem with TCP and expects it to worsen in ad hoc networks where media contention is greater.

It is questionable as to whether BlockAcks may exacerbate ack compression. When TCP receivers experience large numbers of back-to-back TCP segments in blocks, the subsequent TCP acks will also be generated and transmitted in blocks. Rather than the natural multiplexing that occurs in current DCF, where after every two TCP data segments received a TCP ack is transmitted, with large block sizes, many data packets will be received which may be replied to by blocks of back-to-back acks. The loss of multiplexed TCP data frames with TCP acks may increase ack compression and degrade end-to-end performance in high bandwidth high RTT environments. TCP senders receiving compressed TCP acks can either dump large numbers of packets onto the network, suddenly increasing router buffers and the likelyhood of queuing losses, or, pace the transmission of data packets more evenly and smoothly, but at the price of additional delay. This issue is worthy of further study. To conclude, 802.11 BlockAcks may make transmission 10% more efficient [105, 104], however, these efficiency gains may come at the cost of other network goals.

Readers that are concerned with why 802.11n frame aggregation was not discussed alongside 802.11e BlockAcks are encouraged to read appendix B. This appendix describes why the problems of acknowledgment inefficiencies, solved by BlockAck, and too many small packets, solved by frame aggregation, are different problems. Appendix B describes the solutions to too many small packets.

4.3 TCP Improvements

Improving wireless performance at the end points are ideal implementa-

tions because only end host devices need to be updated to improve capacity. The caveat is that end-to-end improvements can not just be wireless specific, improvements must, at the very least, not disadvantage wired end-to-end transmissions. Because of these requirements, and also the requirements for compatibility, end-to-end improvements tend to be evolutionary changes. Currently, the standard end-to-end mechanism for TCP is NewReno with SACK.

4.3.1 NewReno

The problem with wireless packet loss is not reliability, TCP ensures reliable delivery end-to-end, but that losses are treated as congestion. Lost packets cause duplicate acknowledgments to be returned to the TCP sender. Upon reception of three dup acks, TCP New Reno will halve the congestion window. The reason that TCP operates in this manner is because in traditional wired networks, congestion is the primary cause of packet loss. However, in wireless networks, this assumption is no longer valid. In wireless networks, interference and collisions are the primary cause of packet loss.

4.3.2 SACK

SACK or selective acknowledgments [115] are a TCP extension that enhance recovery when multiple packet losses occur within a RTT. Given it is anecdotally recognized that wireless networks experience short bursts of packet loss, multiple packets are commonly lost in the same RTT. This is problematic for pre-SACK TCP because cumulative acks can only hold the

information of the first lost packet. SACK can specify which blocks of data, following the loss, have been successfully received. By informing the sender of which packets have been received and which packets must be resent, multiple packets can be recovered in one RTT. SACK has proven to be beneficial [116] for error recovery in wireless networks and thus has been implemented in all major OSs. It is used in conjunction with both NewReno and Westwood.

4.3.3 Westwood

TCP Westwood [117, 118] was created as an alternative to TCP Reno and is specifically designed for wireless networks. It mimics TCP Reno operation with exponential growth during slow start and linear growth during congestion avoidance phases. Based on the packet sizes being transmitted and RTT estimates, Westwood uses a series of equations to estimate the bandwidth usage of the link. When a packet is lost, the window is reset to the bandwidth estimate rather than halved. Further details of the mathematical derivation of the bandwidth estimates can be found in [117, 118].

TCP Westwood was created to improve the performance of TCP over wireless links. The results of standards based 802.11 and NoAck 802.11 presented earlier were based on the default TCP algorithm, TCP NewReno. This section compares the results of TCP NewReno with TCP Westwood which is specifically designed for wireless networks. Given that Westwood reacts differently to packet loss, an experiment was devised to compare Westwood and NewReno.

The testing topology used was identical to earlier experiments and, for reference, is duplicated in Figure 4.8. The TCP receiver was a Linux laptop

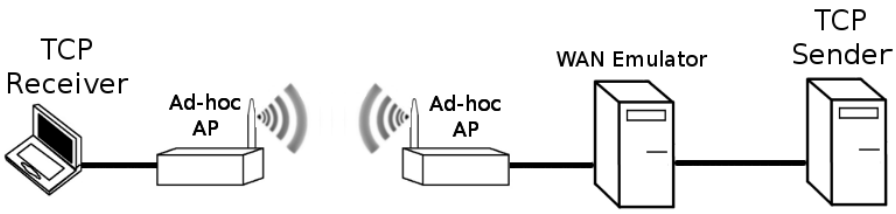


Figure 4.8: Testing setup

running kernel 2.6.27 with TCP NewReno and SACK enabled. The TCP sender was a Linux server running kernel 2.6.18 also with TCP NewReno and SACK enabled. The TCP congestion control mechanism can be midified in linux by editing /proc/sys/net/ipv4/tcp_congestion_control. The bandwidth was averaged by downloading a 144MB file which was downloaded from the TCP sender running an Apache 2.0 web server. The WAN emulator added between 0 ms and 200 ms of delay. Interference was minimised through the use of the 5GHz band. Furthermore, the results shown are derived from heavy averaging of many test runs over multiple days. The performance of TCP Reno and TCP Westwood in standard 802.11 and NoAck 802.11 networks was compared. Two experiments were performed. One using a single TCP flow, the second experiment tested performance with five TCP flows.

The results, shown in Figure 4.9 and Figure 4.10, suggest that Westwood slightly outperforms TCP NewReno in lossy wireless 802.11 networks. However, it is clear from the figures that neither Westwood or NewReno were designed to operate at the loss rates experienced with NoAck 802.11 networks. At this point it is important to acknowledge that losses also occur in standard 802.11 networks too. However, link layer recovery ensures that these losses are many magnitudes less frequent than in NoAck 802.11 networks. Research

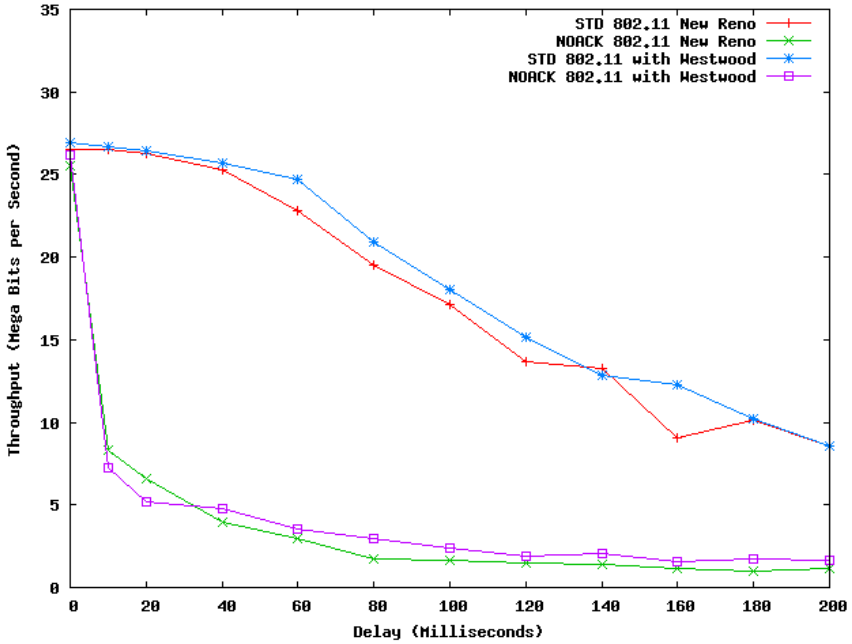


Figure 4.9: Westwood single flow

professing Westwood's superior abilities in high loss networks is referring to loss rates in wireless networks with link layer recovery. The results shown suggest that transport protocols and enhancements, even those designed for lossy wireless networks, are incapable of withstanding losses in NoAck 802.11 networks.

4.3.4 ELN

As TCP (miss)interprets packet loss as congestion, many [119, 120] have postulated that if packet loss and congestion could be signaled differently, then TCP could make the appropriate window adjustments. In the case of congestion, the TCP sender could slow down, else if, interference or collisions were the cause of packet loss, the TCP sender could maintain the conges-

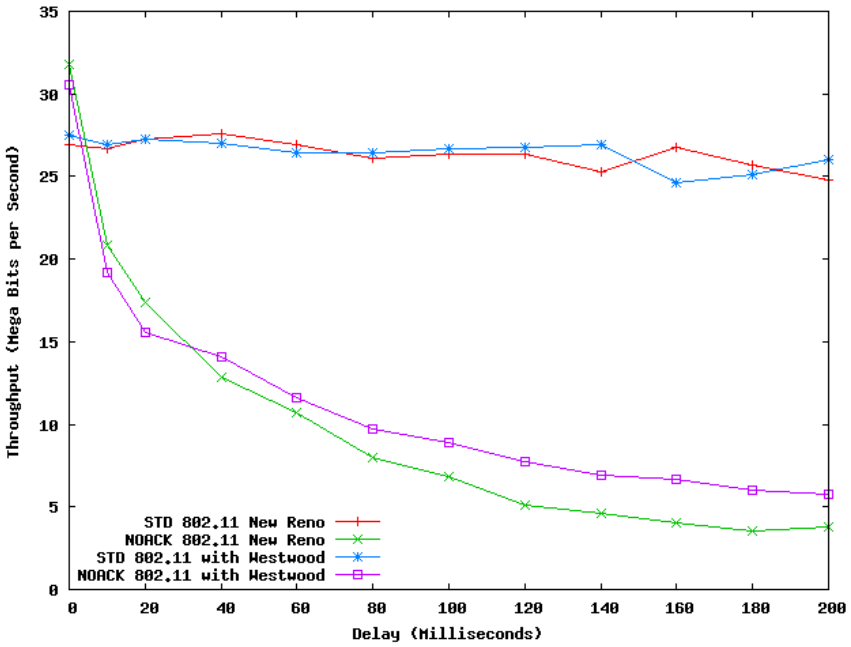


Figure 4.10: Westwood multi flow

tion window. TCP mechanisms that inform the TCP sender that a packet was lost due to interference or collisions are known as ELN (Explicit Loss Notification).

It has been suggested [120] that some wireless losses could be interpreted at the TCP receiver. If a packet with a corrupt CRC (Cyclic Redundancy Check) is received by a TCP receiver, it could inform the sender that the packet was corrupt and therefore not caused by congestion. This solution is problematic because if the CRC is incorrect, it is also likely that either; the TCP header is corrupt and unidentifiable or, the packet will be dropped before reaching the TCP receiver due to a corrupt CRC in the IP header.

Base station ELN implementations are another alternative [119]. If wireless base stations could detect packet loss, they could inform senders using ICMP messages or by tagging returning TCP acks. The difficulty is how to

determine whether a transmission was successful. Base station implementations of ELN typically monitor the success of ARQ acknowledgments. If losses are already being detected at the MAC layer, there is a strong argument that it should be recovered at the MAC layer and thus these schemes have limited use. Furthermore, if a base station is being used to detect loss and send messages to end hosts then it violates the layering principle. This work does not oppose cross layer enhancements, however, if a mechanism is using such techniques, dedicated PEPs, designed to recover packets rather than just inform of the reason for the loss, should also be considered.

4.3.5 Transport layer improvements discussion

TCP Westwood cannot accommodate the levels of loss experienced with unacknowledged 802.11 communication. Furthermore, ELN implementations are of limited use and SACK is also, like Westwood, not designed for the loss levels of NoAck 802.11. Consequently, the end-to-end approaches are incapable of solving this problem. The next section explores cross layer solutions by investigating PEPs (Performance Enhancing Proxies) which locally retransmit packets based on transport layer information.

4.4 PEP

Performance Enhancing Proxies (PEPs) are designed to mitigate link related degradations and are discussed in a dedicated RFC 3135 [121] as well as RFC 3449 [111]. These documents discuss a range of proxies, however this thesis is specifically concerned with proxies that are capable of minimizing

the affects of packet loss on the TCP sender. This section reviews split TCP and the Snoop proxy. A new distributed PEP, designed for the goal of ack efficiency in 802.11 multi hop ad hoc networks, is also devised.

4.4.1 Split TCP

Split-TCP PEPs [122, 123, 124], segment a TCP connection by capturing SYN and SYN-ACK packets. This enables the proxy to imitate each side of the transaction. One advantage is a large reduction in TCP perceived RTTs. Real end-to-end latencies will be the same, however, by splitting the link, the TCP sender can receive acks more quickly building the congestion window faster and thereby completing faster. Split-TCP can also be used to hide losses by placing the PEP between the wireless link and the TCP sender. If the latency between the PEP and the TCP receiver is small enough, losses on the wireless link will have a inconsequential affect on the TCP window as the window size required to fill low latency connections is minimal.

Unfortunately, Split-TCP has numerous problems. Firstly it breaks TCP end-to-end semantics. This means that when a TCP sender receives a TCP ack, the actual packet may not have reached the real TCP receiver. If packets are being buffered on the PEP and the path to the actual TCP receiver breaks, unforeseen application layer problems may arise. Split-TCP is incapable of accommodating route changes between the TCP sender and the PEP. Finally, numerous security protocols will not work through Split-TCP [121]. Despite providing good performance [124], Split-TCP is not recommended as a general solution to lossy wireless links [121].

4.4.2 Snoop

The Snoop proxy [125, 126, 127, 128, 129], detects losses by monitoring TCP acknowledgments. When Snoop sees TCP dup acks returning to the TCP receiver, it will filter these dup acks and retransmit the lost data packet. After the lost packet is replayed, the sequence can continue. If Snoop has filtered all the dup acks, the TCP sender should have no knowledge of the loss. Snoop proxies do not break TCP end-to-end semantics but, as a result, are dependent on their ability to hide link losses from the sender.

Figure 4.11 shows the operation of snoop proxies. In this example, packet 1448 and 2896 were received successfully, however, when a second and third ack for packet 2896 is received it is assumed that packet 4344 was lost. The snoop proxy then checks its cache and performs a retransmission. The snoop proxy determines that packet 4344 was lost because:

$nextseqnum = currseqnum + currpayloadsize$

The problem is that a large number of duplicate acknowledgments must often be filtered. Also, it is not uncommon for Snoop to resend a lost packet, only to have the retransmission corrupted too. Given the number of packets transmitted per second, such scenarios occur more regularly than expected. Snoop has no mechanism to determine whether a lost packet, that was recently replayed, has also been lost. Thus, it is unknown how many dup acks should be filtered before resending the missing packet a second, third, fourth time. Furthermore, given that these duplicate acks may contain SACK information about other lost packets, it is unknown how often lost packets, specified by SACK, should be resent.

A number of studies suggest interoperability problems between SACK

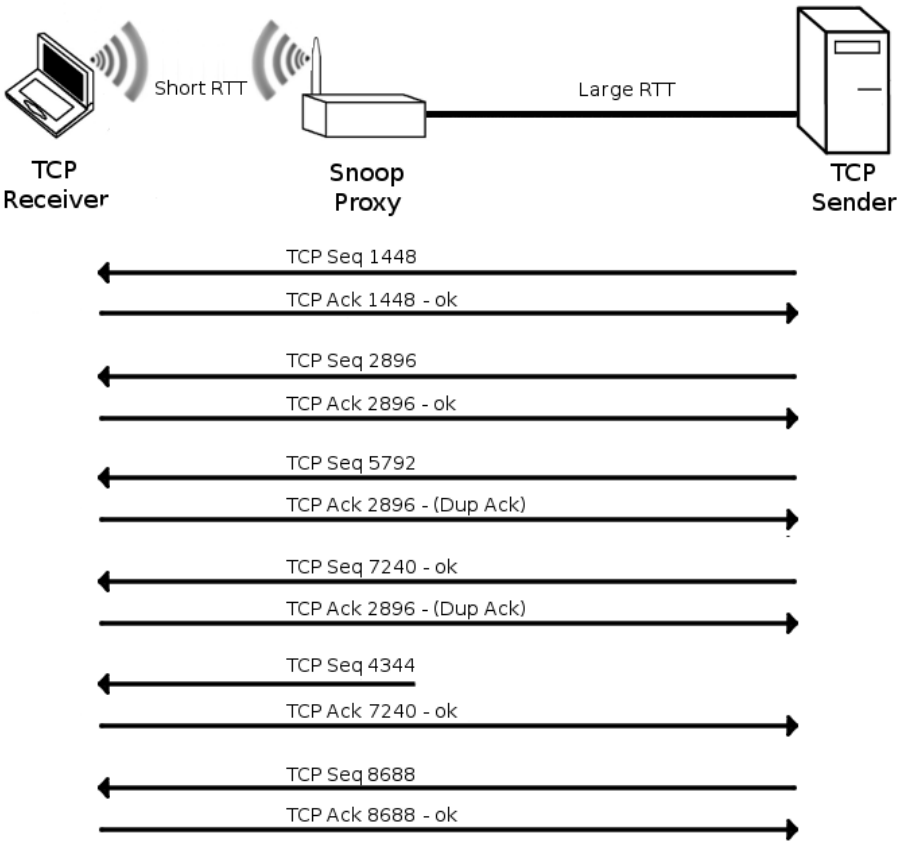


Figure 4.11: Snoop proxy operation

and Snoop [129, 128, 130]. These studies suggest that Snoop is unable to completely hide losses from the TCP receiver, especially in the presence of burst losses. Finally, it is unknown how many dup acks should be filtered. Eventually, filtering dup acks will cause TCP timeouts. Ack compression also becomes problematic because the retransmission of one missing packet may result in the next unfiltered ack, acknowledging a huge number of data packets. While Snoop can improve performance in lossy networks, numerous problems impinge performance [129, 128, 130]. In testing Snoop, the lack of any working implementation for a modern OS required a rewrite of Snoop. The code can be found at http://www.bridgingthelayers.org/.

4.4.3 D-Proxy

D-Proxy is a new proactive distributed TCP proxy designed to overcome the limitations of Snoop and Split-TCP. D-Proxy is distributed because it uses a proxy either side of the lossy link. It is proactive because, instead of waiting for TCP acks to confirm packet loss, it analyzes TCP data sequence numbers. If the sequence number received is greater than the sequence number expected, it is assumed that there may be one or more missing packets. A message is then sent to the proxy on the reliable side of the link to request that the packet be resent. Figure 4.12 shows the basic operation of D-Proxy. Note that the missing packet was discovered because 5792 was sent when 4344 was expected. When a loss is detected, D-Proxy buffers frames until the lost segment can be replayed and reorganizes them such that they are forwarded in sequence. The expected sequence number is simple to predict

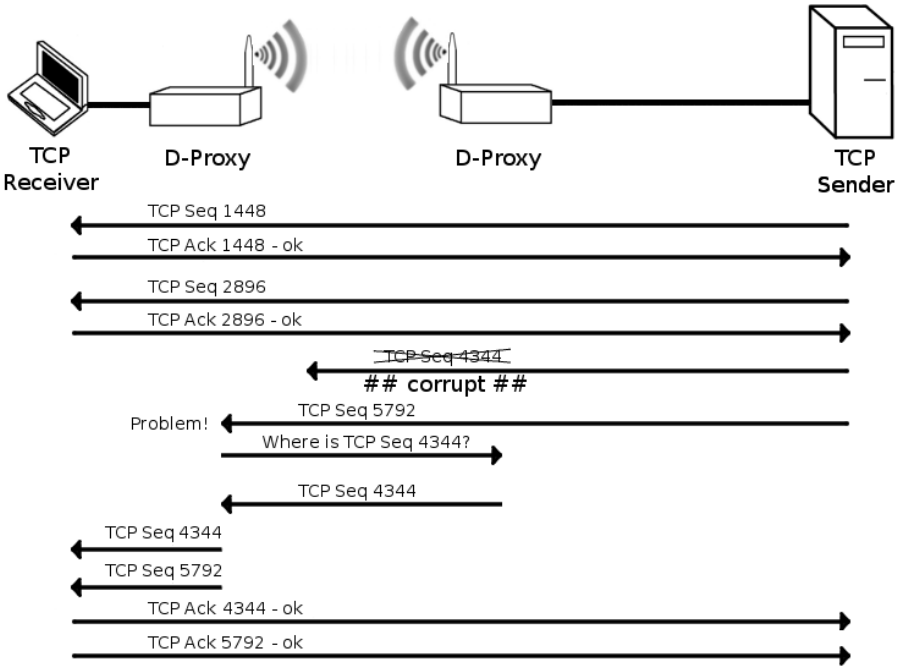


Figure 4.12: D-Proxy simplified

because it is equal to the current sequence number plus the payload size.

While the basic concept of D-Proxy is relatively simple, the implementation required significant work. D-Proxy maintains TCP state information and each flow is differentiated based on source IP, destination IP, source port and destination port. The individual packets being cached are identified within their flow based on sequence number. D-Proxy was implemented in Linux using the ip_queue library which passes packets from kernel space to user space for processing.

4.4.3.1 Inter proxy communication

D-Proxy buffers TCP flows until lost packets can be recovered and reordered. However, like Snoop, large delays, perhaps caused by buffering

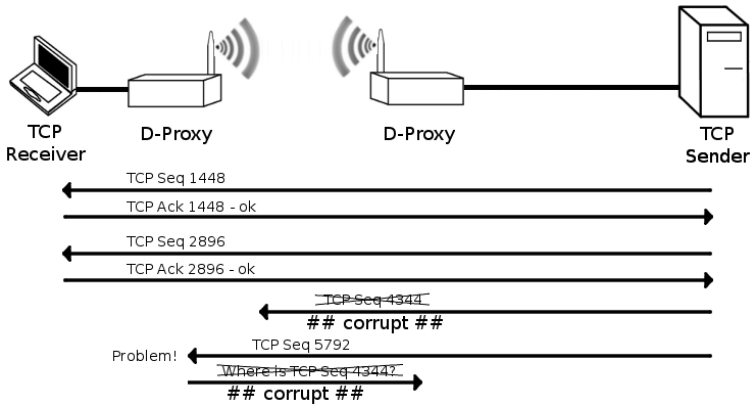


Figure 4.13: D-Proxy: losing a data packet and the retransmission request

packets for too long, will cause TCP timeouts. Therefore, the speed at which D-Proxy can recognize a loss, request retransmission of the packet and resend it is of utmost importance. It is anecdotally recognized that wireless losses occur in bursts. This is problematic because it increases the likelihood that when a packet is lost, the request that is sent asking for its retransmission may also be lost. It is equally likely that the actual data segment being retransmitted might be lost. These problems referr to the scenarios shown in Figure 4.13 and Figure 4.14. Therefore, the mechanism used to request the retransmission of lost packets is critical.

The implementation used UDP sockets to re-request lost packets as they are significantly faster than TCP sockets. UDP may appear an odd choice because the reliability of retransmission request messages is critical, however, TCP error recovery is far too slow to be useful in these circumstances where errors must be detected and recovered on a millisecond time-scale. For this reason, UDP was used to implement D-Proxy's fast reliability mechanisms. This mechanism had to provide fast two way reliability for both the UDP retransmission request and the actual data segment. Two mechanisms were

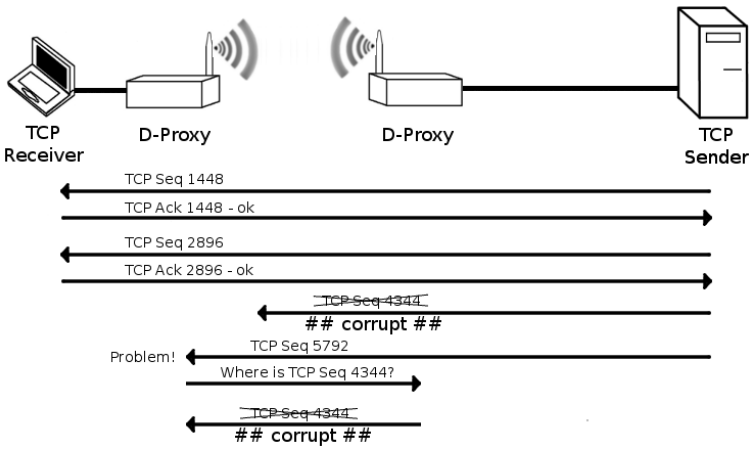


Figure 4.14: D-Proxy: losing a data packet and the retransmitted data segment

used to infer that either the UDP retransmission request, or, the retransmitted TCP data segment had been lost.

4.4.3.2 Timeouts

One mechanism was time based. The amount of time required to recover missing packets is continually averaged. Using a continual average of retransmission times is designed to adapt to natural link latency as well as link load. Packets will be re-requested after two, three and four times the average retransmission time. On the fourth re-request, the buffer for that flow will no longer be held awaiting that packets retransmission.

The amount of time to wait for a packet to be resent before requesting retransmission is a trade-off. If a replayed packet is not lost but merely waiting in buffers, or waiting for media access, sending further retransmission requests will only exacerbate delays and create superfluous retransmissions. Equally, waiting too long before sending further retransmission requests only

increases the number of packets being buffered and the likelihood of TCP timing out. Using UDP and a timeout based retransmission mechanism provided performance benefits, however, performance was still suboptimal. It became clear that the process of reliably recovering packets, needed to be quickened. The problem is the same that was experienced in Snoop. When a lost packet is resent; how can the proxy determine whether the retransmitted segment was successful? The problem, shown in Figure 4.13 and Figure 4.14 occurs frequently because of the burst loss nature of wireless networks.

4.4.3.3 Retransmission order

The key mechanism used to provide greater reliability to both the UDP retransmission request and the actual retransmitted TCP packet is more complex but provides extremely fast and reliable error recovery. Its operation is best understood with an example. Figure 4.15a shows the buffer on the unreliable side of the wireless link. The first three packets, 1448, 2836 and 4344 are all in sequence and are passed back to the kernel for transmission. Following 4334 there are holes between 4344-8688 and 8688-11584. Based on the premise that the next sequence number equals the current sequence number plus the packet size, D-Proxy can ascertain that 5792 is missing and also that 10135 is missing. Due to the size of the hole between 4344-8688 and the packet sizes in the flow, D-Proxy assumes that 7240 is also missing. Each missing packet generates its own retransmission request. So, three individual retransmission requests will be sent containing a TCP flow ID and the missing sequence number.

In Figure 4.15b the replayed packet with sequence number 7240 is re-

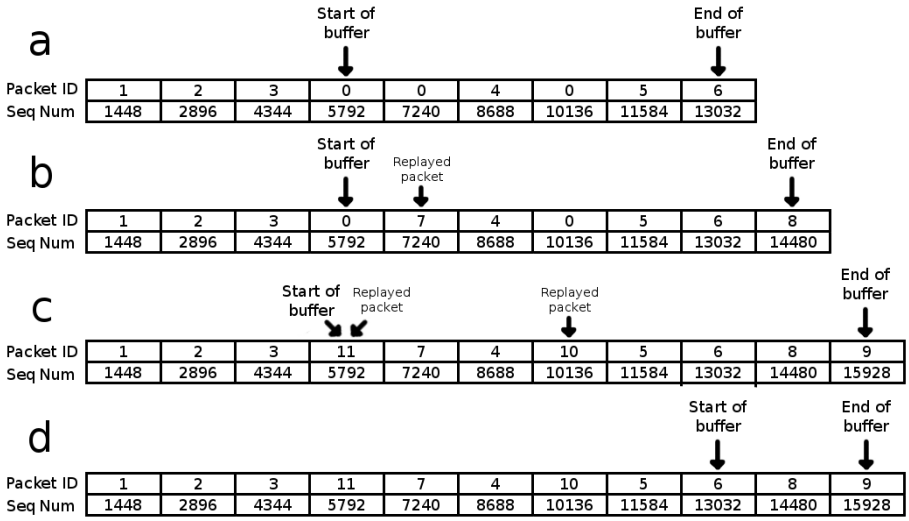


Figure 4.15: Packet recovery based on retransmission order

ceived and then a new packet 14480 is received. The fact that the replayed packet 7240 was received before 5792 is used as an indicator that either the UDP request or the replayed 5792 packet was lost. As stated earlier, each missing packet is requested with an individual UDP packet thus the individual retransmission requests are sent in order. If a retransmitted segment such as 7240 is received ahead of an unfilled hole such as 5792, it indicates that either the retransmission request for 5792 or the actual retransmitted data segment was lost. If packet loss was randomly distributed then this mechanism would be less effective, however, as packet losses frequently occur in bursts, this method of packet recovery was many magnitudes faster than a timeout based system.

Similar to Figure 4.15a, at the current point in time, Figure 4.15b, no more packets are being passed back to the kernel for transmission. It is critically important that packets are transmitted in order to prevent duplicate

acknowledgments causing the sender to half the congestion window.

In Figure 4.15c, a new packet, 15928, arrives and packet 10136 is replayed. Finally, packet 5792, which had to be re-requested for a second time is replayed. With 5792 now in sequence, the packets can be sent to the kernel for transmission.

4.4.3.4 Staggered TCP catch-up

In Figure 4.15d, note that the buffer has only moved to packet 13032. The avoidance of packet bursts following a recoverd segment is of utmost importance. For example, a large number of packets can be queued waiting on the retransmission of a single packet. When the missing packet is replayed, it is desirable to avoid immediately replaying 30 or more consecutive packets from the same TCP flow as this could be to the detriment of other flows. Instead, a maximum of five packets from that specific flow is sent. Following these five packets, the input buffer is rechecked. If the next packet on the ip_queue input buffer is a packet from a different flow, then that flow will be serviced. If the next packet on the input is also part of this flow then another 5 packets can be sent from the buffer to the kernel. The main purpose of this mechanism is to stagger the catch-up that occurs when filling the holes of a TCP sequence such that other TCP flows are unaffected. This aids fairness and prevents ack compression.

4.4.3.5 Variable per flow buffer size

Similar to normal router buffers, more buffering increases latency, slowing

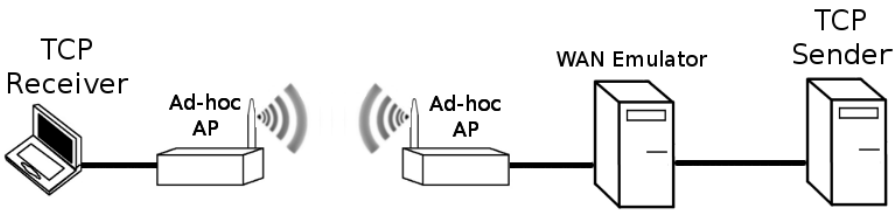


Figure 4.16: Testing setup

the reaction to congestion. Buffering is important for D-Proxy because larger buffers increase the opportunity for missing packets to be re-requested, resent and reordered for delivery. D-Proxy has a buffer size of 150 packets. With one TCP flow, the entire 150 packet buffer will be utilized. The addition of more flows, will divide the buffer equally. Therefore, if there are 5 flows, each flow will have a buffer of 30 packets. Flows may exceed their buffer size. For example: there is one flow utilizing a buffer of 150 packets and then 2 additional flows are quickly added, reducing the buffer size to 50 packets per flow. The 100 packets that exceed the buffer size from one flow are not lost or dropped, instead the oversize buffer will continue being processed, however, no packets that exceed the allowed buffer size will be waited on to put be back in sequence. Retransmission requests will have already been sent for the holes, but, D-Proxy will no longer wait to reorder packets that are exceeding the buffer size. Receiving the lost packets out of order fortuitously causes the TCP sender to slow down to accommodate the two new flows. The D-Proxy code can be found at http://www.bridgingthelayers.org/

4.4.3.6 Experiment

The testing setup used to obtain these performance results is shown in

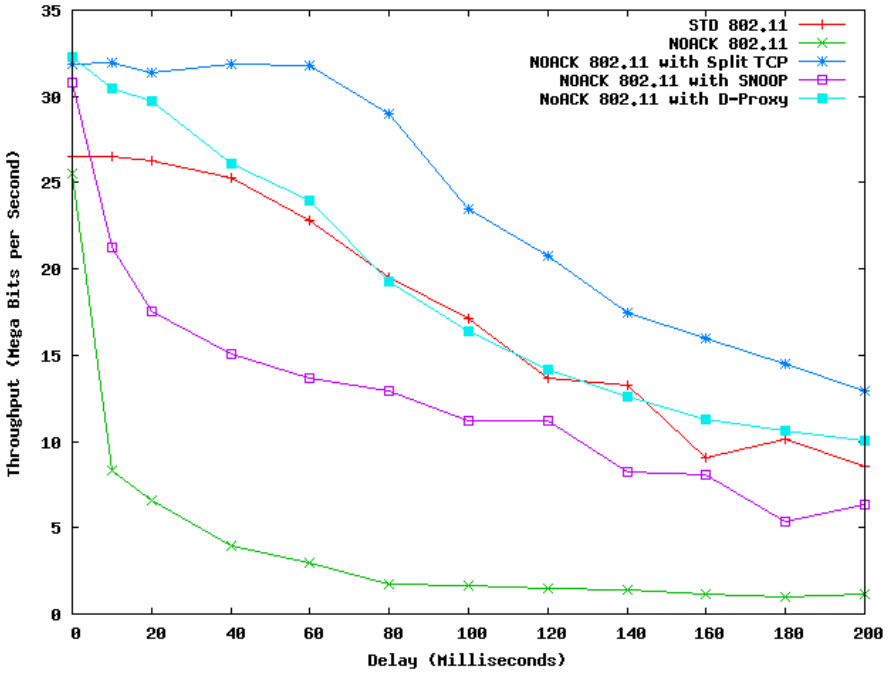


Figure 4.17: Reliable 802.11 link with a single TCP flow

Figure 4.16. A series of bandwidth tests were repeatedly performed and averaged with a range of introduced WAN latencies between 0ms and 200ms. Single flow tests were performed whereby a single download of a 144MB file from an Apache web server was performed. There were also multi flow tests where five concurrent downloads of the 144MB file were performed. These bandwidth tests were performed over a 54Mb/s 802.11a wireless link. The reliable scenarios featured a download over a link with a strong SNR. The unreliable tests were created by locking the network card's speed at 54Mb/s and moving the APs past the point where they would normally rate shift. The results of the reliable and unreliable single flow tests are shown in Figure 4.17 and Figure 4.18. The results of the reliable and unreliable five flow tests are shown in Figure 4.19 and Figure 4.20.

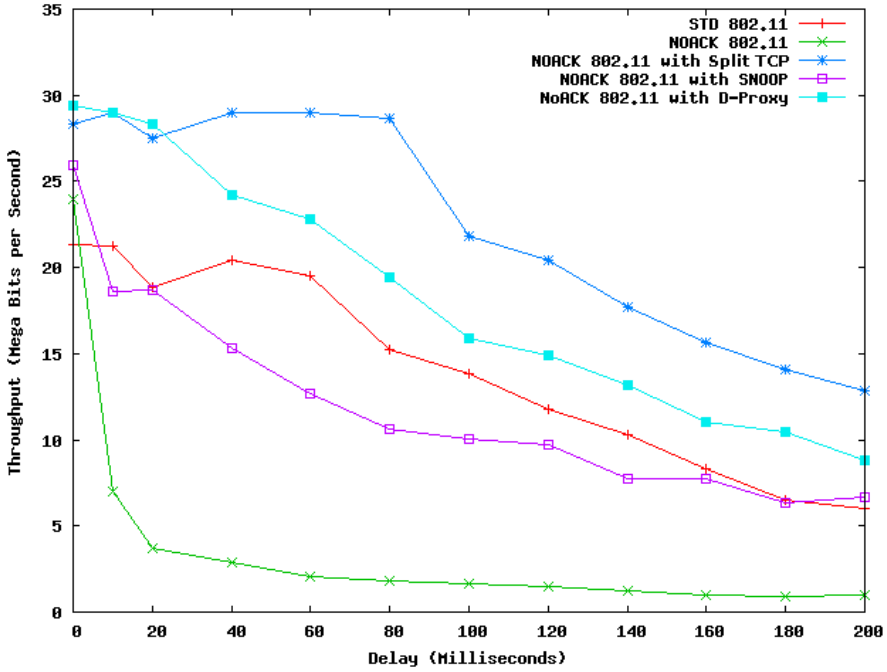


Figure 4.18: Unreliable 802.11 link with a single TCP flow

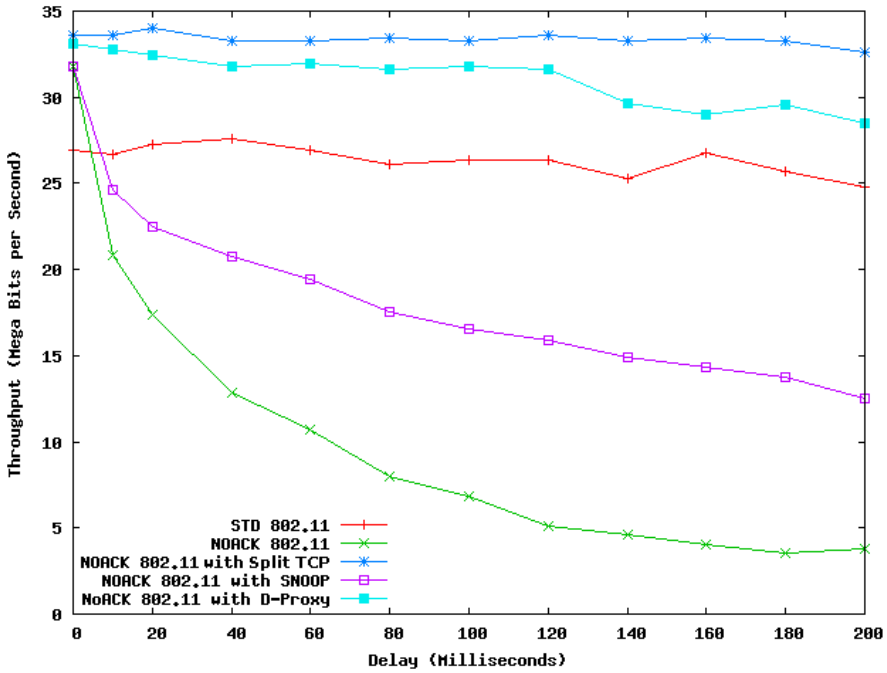


Figure 4.19: Reliable 802.11 link with five TCP flows

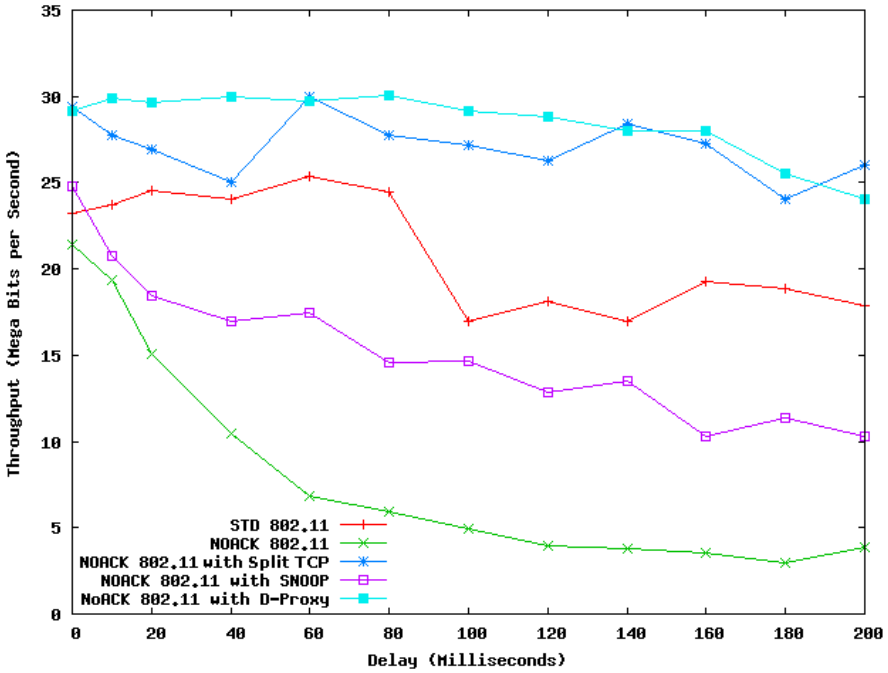


Figure 4.20: Unreliable 802.11 link with five TCP flows

4.4.3.7 Snoop Results

Snoop proxies can increase performance in standard 802.11 networks, however, both this and numerous other recent studies [129, 128, 130] have suggested that Snoop is unable to hide losses from TCP senders. The results suggest that Snoop cannot hide the heavy packet losses in NoAck 802.11. Furthermore the implementation of Snoop in multi hop ad hoc networks is problematic. For example, along a five hop wireless path, a packet could be lost at any stage. If packet loss occurred on the first hop, the packet will not be recovered until the duplicate acknowledgment was re-received on the first wireless router. This will substantially delay the detection of loss and increase the incidence of TCP timeouts.

4.4.3.8 Split TCP results

The Split TCP results show that the performance of Split TCP with NoAck 802.11 exceeds standard 802.11. With a Split TCP proxy deployed as close as possible to the TCP receiver, the minimal RTT on the unreliable wireless side of the proxy means that large congestion windows are unnecessary. On the long RTT side of the proxy, the wireless losses are shielded from the real TCP sender. The use of Split TCP in this manner achieves the goal of a negatively acknowledging system as dup acks are the only feedback mechanism used to indicate loss.

As Split TCP has been in use for over a decade, it is important to question why it is not standard in WiFi implementations. Split TCP is only recommended for specific cases because it breaks TCP's end-to-end semantics. When a TCP ack is received for a TCP data segment, the TCP sender assumes that the data has been received correctly. Thus, Split TCP may ack segments before the real TCP receiver has received the data. If a link failure occurs between the TCP receiver and Split TCP, the TCP sender may have received acknowledgments for data that was never actually received. This is the main argument against Split TCP; it breaks TCP's end-to-end semantics. Many security protocols such as SSH do not work through Split TCP. A more thorough treatment of these issues is given in RFC 3135 [121].

Implementations for ad-hoc networks are also problematic. Mindful that low RTTs are quintessential to performance over the unreliable link, in multi hop ad hoc networks, many Split TCP proxies will be required throughout the network. Due to the number of Split TCP nodes pretending to be end-

points, end-to-end transactions will fail with a single path/routing change.

4.4.3.9 D-Proxy results

The inapplicability of Split TCP and Snoop proxies in multi hop ad hoc networks, provided the impetus for a fresh approach. D-Proxy was started with the notion that an ideal proxy should not break end-to-end TCP semantics but should also be more proactive than Snoop in its approach to detecting packet losses. The solution, D-Proxy, has been shown to provide superior performance to standard 802.11 without the drawbacks of Split TCP or Snoop. The results show that D-Proxy performance is equivalent to Split TCP.

D-Proxy does not break TCP end-to-end semantics but does hide link losses from the TCP receiver and can be implemented in multi-hop ad hoc networks. There are two current drawbacks of D-Proxy. Firstly, network layer security mechanisms such as IPSec will have to bypass the proxy. Secondly, in the current implementation, CPU utilization for the packet caching side of the proxy is too high to run on embedded systems. This problem may be solvable by implementing D-Proxy as a kernel module.

4.4.4 Theoretical comparison with BlockAck

This chapter claims that D-Proxy is superior to the other PEPs. It also shows that the performance of D-Proxy with NoAck 802.11 far exceeds standard 802.11. The question which remains; is performance superior to the 802.11e BlockAck function? Unfortunately, at the time of testing, the driver

used did not have a working BlockAck implementation so real life comparisons were impossible. Simulations claim that 802.11e BlockAck can improve efficiency in bulk transfers by 10% [105, 104]. The real world wireless tests reveal that the difference between the the peak transfer capacity of standard 802.11 and NoAck with D-Proxy is 21.5%.

The key difference between the two schemes is that BlockAck is a positively acknowledging proxy. While BlockAck undoubtedly increases efficiency, it still provides reliability by acknowledging packets that have been received. D-Proxy is a negatively acknowledging proxy that uses TCP sequence numbers to decipher what is missing. The drawback of D-Proxy is complexity; requiring a proxy on both ends of a link and, in addition, caching and buffering packets. However, BlockAck also operates in a similar fashion, buffering packets at both sides of the link.

Another advantage is that unlike BlockAck, D-Proxy will not group TCP data segments and TCP ack packets into blocks, instead, D-Proxy will maintain the natural multiplexing of two data packets for every TCP ack, steadying the delivery of TCP acks and preventing ack compression. When packets have been buffered at D-Proxy waiting for a retransmitted segment, the emptying of the buffer is staggered to prevent packet bursts from a single TCP sender getting exclusive priority.

D-Proxy is also capable of allowing multiple packets from different senders to pass through the proxy while waiting for retransmissions. This is because D-Proxy groups packets based on TCP flows, not MAC layer senders. This means that the loss of one packet will not require all subsequent packets to be buffered because D-Proxy maintains a separate buffer for each TCP flow.

D-Proxy also performs a stock-take of what packets are present and what

are missing after every packet whereas BlockAck performs this function periodically or at the end of every block of packets. This means that BlockAck may have delayed recognition and reaction to the loss. Furthermore, D-Proxy does not require lengthy medium reservation and thus D-Proxy may facilitate fairer QoS in the presence of multiple transmitting nodes. I predict that D-Proxy is more TCP friendly and will offer better performance, fairer QoS, and faster reaction to lost packets than BlockAck. When drivers become avaliable, a real world comparison will be performed.

4.5 Conclusion

This study started by questioning the need for 802.11 acks. It was shown that without 802.11 acks, links with RTTs greater than 1ms were underutilized by TCP because packet loss, being interpreted as congestion, caused TCP window reductions. The IEEE standards based work attempting to address this problem was then explained and explored. End-to-end approaches such as Westwood were also experimentally tested, however, the poor results merely reinforced the need for local recovery of lost packets. Experiments with PEPs revealed that Split TCP provided excellent performance, but had a number of limitations and implementation problems for ad hoc mesh networks. It was shown that Snoop proxies were not suitable for hiding the heavy losses caused by unacknowledged 802.11. This prompted the creation of a new PEP specifically designed to address the limitations in previous proxies. D-Proxy is a distributed PEP designed to discover and recover losses over a wireless link. It is a proactive TCP proxy because it recovers link losses before they reach the TCP receiver. D-Proxy is capable of nega-

tively acknowledging packets and is implementable in ad hoc networks. By removing acknowledgments using the 802.11 NoAck function, D-Proxy can increase performance by 22%. Currently, CPU requirements for D-Proxy are too high to operate in embedded systems, however, in the future, I plan to rewrite this proxy in attempt to drastically reduce the CPU requirements.

Chapter V

Conclusion

The OSI networking model was designed predominately for wired networks in 1994 [7]. It has been a highly successful networking model and many believe it to be responsible for the proliferation of an open and interoperable Internet [8]. Multi hop ad hoc networks are fundamentally different from the well planned, reliable, administrator configured, wired networks for which the OSI model was designed. The underlying technologies of multi hop ad hoc networks necessitate that the guidelines of the traditional layered networking model be re-examined. Consistent with other work in this field, this thesis uses a cross layer approach to increase the scalability and performance of real world multi hop ad hoc networks. This PhD has made three major contributions in the areas of routing, multi hop contention and acknowledgment efficiency.

Traditional routing protocols were never designed for the self-forming, self-healing properties required in multi hop ad hoc networks. Hierarchical addressing, which occurs alongside routing at the network layer, is inappropriate for these networks. The self-forming, self-healing requirements of multi hop ad hoc networks necessitates a flat addressing structure. While this may indicate that the data link layer is more suitable for routing, whichever layer

is used, the original design intentions of the OSI layers will be blurred. In addition to the layering and addressing problems, chapter 2 details many other features that make traditional routing protocols inappropriate. The solutions to these problems are described with examples from the literature. A real world experiment is performed to assess the key techniques and determinants of performance. The study shows that in small networks, the Babel routing protocol provides higher throughputs than both BATMAN and OLSR. It also shows that in small networks, the network overhead of the routing protocol may be the biggest determinant of performance and that the OSI layer of the routing protocol has no major performance implications. Future work could investigate whether these results apply to larger experimental testbeds.

802.11 is an ideal technology to use with multi hop ad hoc networks because it is an open standard, unlicensed, mass produced, the focus of significant research efforts and is embedded in an enormous number of technologies. Unfortunately, when it was originally designed, it was intended as a one hop technology to connect wireless devices, such as laptops and PDAs, to the wired network via a centralized AP. Subsequently, the contention problem, which is responsible for performance degradations when multi hopping transmissions using the same channel, were never considered. Some studies propose new multi channel MAC layers to make use of multiple channels, allowing multiple concurrent transmissions. Given that a major benefit of 802.11 is its economies of scale, which enable a highly sophisticated technology at consumer grade price, breaking 802.11 standards compliance is undesirable. Chapter 3 argues that by using multiple ad hoc 802.11 radios, assigned to different channels, contention problems can be alleviated without breaking standards compliance. This approach requires a channel selection

mechanism that can dynamically and intelligently assign channels to radios. In addition, it also requires routing protocols to incorporate bandwidth and eventually channel based metrics for multi radio nodes. Chapter 3 discusses the development of, RDCS, a channel selection mechanism that operates with the OLSR routing protocol. It showed that multi hop performance in multi radio networks could be increased with this scheme. To further routing driven channel selection, it should be applied to other routing protocols such as Babel or BATMAN. Furthermore, as a result of RDCS's reliance on the routing protocols metric, performance increases in RDCS will be realized when bandwidth based metrics such as ETT or channel metrics such as MIC can be incorporated into routing protocols.

TCP interprets all packet losses as congestion because it was never designed for lossy wireless links. When TCP misinterprets packet losses as congestion, it reacts harshly by reducing the TCP congestion window; throttling the connection. Losses due to interference or collisions occur frequently in wireless networks necessitating link layer reliability mechanisms to hide losses from TCP. While these mechanisms operate adequately, they introduce a large overhead. Chapter 4 proposes that if these acknowledgments could be removed, large performance increases would be possible. It investigates alternative reliability mechanisms such as 802.11 BlockAck, TCP Westwood, as well as Snoop and Split TCP PEPs. However, these methods are shown to be inadequate or inefficient for multi hop ad hoc networks. This thesis introduces a new mechanism known as D-Proxy which is designed to provide link layer reliability without the overhead. Our novel proxy uses entirely new techniques. D-Proxy is a distributed proactive proxy that analyzes TCP data sequence numbers in attempt to find and fill holes in the transmis-

sion sequence. It is also a negatively acknowledging proxy, sending messages only when a frame has been lost, and thus its overhead is minimal compared with the existing positively acknowledging reliability mechanism. The MAC layer modifications required to remove the existing link layer reliability mechanism can be performed without breaking standards compliance using the the 802.11e NoACK function. The removal of the link layer acknowledgment, combined with D-Proxy, can boost performance by approximately 22%. In the future, I intend to rewrite D-Proxy such that it can be included in embedded operating systems such as Voyage Linux, dd-wrt and OpenWRT. Furthermore, as this technology should theoretically work with any wireless link, the advantages it can provide in other wireless networks such as microwave and WiMAX should also be explored.

Multi hop ad hoc networks are attractive because they can offer greater range, throughput, spatial efficiency and power consumption. Additionally, numerous new, previously infeasible deployment scenarios are enabled. These scenarios include, community networks, developing world communications, disaster recovery and emergency services. These new networks must be self-forming, self-healing, robust networks and are very different from the reliable well planned wired networks for which the OSI model was originally designed. Many of the traditional technologies, designed to operate strictly within the existing OSI networking model boundaries, are inadequate. This thesis has argued that due to the problems caused by traditional protocols, a blurring of these layers is unavoidable. The problems of routing, multi hop contention, and acknowledgment efficiency are all experimentally investigated. The work described in this thesis has furthered understanding in these areas.

Appendix A

RDCS Implementation

A.1 Additional Features

A.1.1 Channel locking

Channel locking, is designed to accommodated different mesh architectures. When users setup their RDCS.conf file they are instructed to create the same number of user defined bands as interfaces. The first interface will be assigned to a frequency in the first band and the second interface will be assigned to a frequency in the second band. By default, with channel locking turned on, each interface will remain in the initial user defined band. This is necessary for setups such as Figure A.1a where a mesh node may be designed with two interfaces, one which only operates in the 2.4 GHz band and the other which operates only in the 5 GHz band. In this situation the user would carefully define the bands to be used by each radio and would want RDCS to retain the original band-interface binding. Equally, a user may be using two 802.11a/b/g interfaces but have 5 GHz and 2.4 GHz antennas connected (Figure A.1 b).

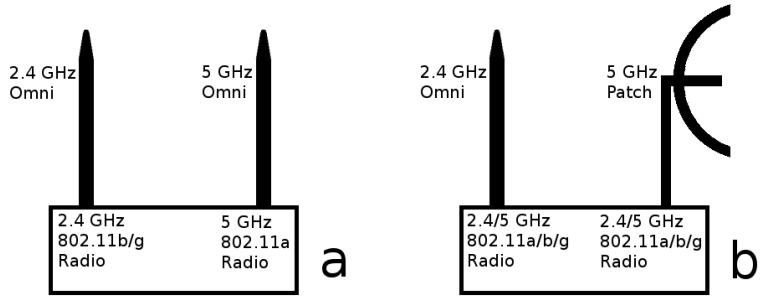


Figure A.1: Architectures that require channel locking

A.1.2 Interoperation with non-802.11 links

This section describes RDCS's Interoperation with non-802.11 links and interfaces. It is likely that many designs and architectures for wireless mesh networks will contain non 802.11-links such as Ethernet, ADSL or WiMAX. Subsequently, channel selection mechanisms should seamlessly integrate with other technologies. Non-802.11 links should be specified in the olsrd configuration file (olsrd.conf) to enable them to be used as a routed link[1], however, they should not be specified in RDCS.conf as they will not be used for channel selection.

To describe the interoperation with non-802.11 links two examples will be provided. Thr first example, shown in Figure A.2, demonstrates neighborhood network whereby a user has a couple of single radio wireless APs which he wishes to connect to a community network. The APs are connected via Ethernet and the user runs both OLSR and RDCS on each node specifying both Ethernet and 802.11 interfaces for OLSR routing and only the wireless interface in RDCS's configuration file. When OLSR starts, the

[1] OLSR works with any technology as it operates at the network/IP layer

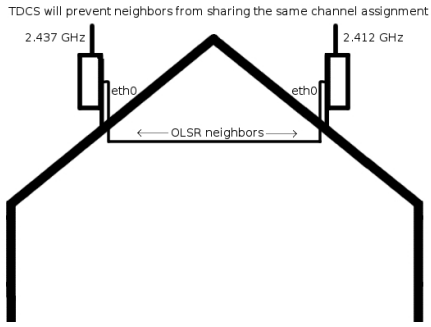


Figure A.2: Non-802.11 interoperation

wired neighbors will immediately be discovered, becoming OLSR neighbors. Subsequently, when RDCS applies the dual link mask (see Section 3.6.4.3), the two wired neighbors will mask each others radio channels. This ensures that while looking for other neighborhood nodes, they will never fall on the same channel. OLSR's reliability metric will also ensure that, even if the wireless radios did fall on the same channel, they would not be installed in the routing Table due to the wired links superior reliability.

802.11 nodes could also be combined with WiMAX. Future mesh networks may employ WiMAX alongside 802.11 for additional backbone connectivity and network resiliency. A advantage of incorporating WiMAX is the frequency separation and diversity gained from using a licensed frequency technology together with an unlicensed technology. Similar to the previous example, WiMAX neighbors will be discovered by OLSR and routes installed in the routing Table. When RDCS evaluates the channel assignment of 802.11 interfaces, the frequencies of the 1-hop WiMAX neighbors will be masked. Similarly to the Ethernet example above, RDCS will prevent WiMAX connected nodes from becoming direct 802.11 neighbors.

This section details and discusses the MadWiFi and OLSR issues that

did not affect the results but did require work around to provide a working system. It is hoped that in the future these issues may be fixed which should dramatically simplify the RDCS code.

A.1.3 MadWiFi Issues

The MadWiFi driver (v9.3.3) required modification to be usable. Under standard ad-hoc operation with MadWiFi the BSSID (Basic Service Set Identifier) is automatically generated from the first node that it connects to. The BSSID is used to filter packets from other nodes that may be transmitting with the same ESSID. This process of discovering the BSSID and using the mac address of the first node seen is flawed in a multi-hop scenario. Consider the scenario where Node A and Node B discover each other and connect, generating BSSIDs based on each others MAC address. When Node C comes along, it connects to node B and uses its SSID to generate a BSSID. However, neither A or B is able to communicate with C because of the SSIDs that have been generated. This problems has become known as BSSID partitioning. This problem was solved by modifying the driver to generate a BSSID based on a hash of the ESSID. This means that all nodes with the same ESSID will have the same BSSID.

A.1.4 OLSR issues

OLSR is the most widely used ad hoc routing protocol in the world. It is the de facto standard for many community ad-hoc wireless networks such as Freifunk [131] and has been thoroughly tested in single radio networks (par-

ticularly in combination with some wireless routers such as OpenWRT [132]). However, during the development of RDCS a number of difficulties were encountered. The majority of routing in Unix based operating systems works by creating a routing protocol routing table which is separate from the kernel routing Table. This routing protocol table is independently maintained and informs and modifies the kernel routing table. In OLSR, routes are added and deleted from the kernel routing table as required, however, due to the channel switching used in RDCS, certain routes are sometimes not removed from the kernel routing table. Although the OLSR routing table has maintained the correct information, discrepancies can form between the OLSR routing table and the kernel IP routing table. The only way to ensure that OLSR updates the kernel IP routing Table with fresh routes following channel changes is to separately maintain the kernel routing table. When RDCS begins, it forks a thread that will maintain the kernel routing table, keeping it in-line with the OLSR routing table and preventing. This is achieved with a function that compares the OLSR routing table and the kernel routing table any reconciles any discrepancies. This issue may have been fixed since v0.5.4.

Appendix B

802.11n frame aggregation

The 802.11n standard [103] aims to provide data rates significantly beyond 802.11a/g's 54Mb/s with throughputs exceeding 100Mb/s. However, as discussed in section 4.2.1, with data rate increases, the relative overheads of 802.11 also increase due to fixed packet overheads such as IFS and preambles. Figure 4.6 was given as an example of how large increases in data rates begin to give marginal throughput benefits given the existing 802.11 DCF MAC. To complement the increased data rates being offered in 802.11n, it was recognized that MAC layer changes were also necessary. By concatenating multiple frames together, time can be saved on IFS and preambles. This is not a new idea, Tourrilhes [133] published a similar design in 1998 and since then, frame aggregation has been used to increase performance in proprietary implementations.

B.1 A-MSDU

There are two forms of frame aggregation in 802.11n, these are known as A-MSDU (Aggregated MAC Service Data Unit) and A-MPDU (Aggregated MAC Protocol Data Unit). A-MSDU can be thought of as micro aggregation

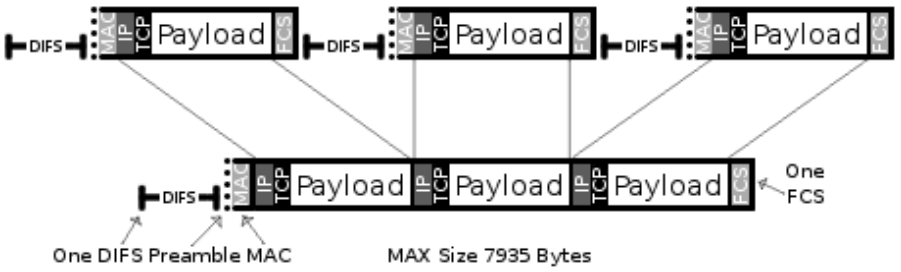


Figure B.1: A-MSDU (Aggregate MAC Service Data Unit)

because it will be used to join multiple small frames into one large frame. Efficiency savings can be made by using one MAC header, preamble, IFS and FCS (Frame Check Sum). The manner in which A-MSDU aggregates frames is shown in Figure B.1. As only one checksum is generated, if a single part of the transmission is corrupted, all of the aggregated frames must be retransmitted. In 802.11, the chance of errors increase as the frame grows so A-MSDU is limited to 7935 bytes [134]. The other limitation is that the frames being aggregated must have the same source address, destination address and QoS class.

B.2 A-MPDU

If A-MSDU is micro aggregation, A-MPDU is macro aggregation as it is designed to merge fully sized frames or A-MSDU's together to further save on preambles and IFS times. Figure B.2 shows that in A-MPDU both MAC headers and the FCS are retained. In addition, a 4 byte MPDU delimiter is added to each frame. The reason for this is because A-MPDU is designed to aggregate up to 64k and thus the individual frames within the aggregated packet must be reliable. Put simply, the loss of one frame within the A-

MSDU should not result in the loss of the entire block.

The mechanism used to provide reliability within the A-MPDU is worthy of detailed discussion. Note that in Figure B.2 a frame delimiter and padding is added to the segment. The MPDU delimiter and specifically the length field within this delimiter is key to A-MPDU's because it allows the receiver to count the number of bytes in each frame. This is used to determine where aggregated frames begin and end. If the FCS of one frame fails, the receiver can skip to the following frame. However, it is likely that if the frame FCS is corrupt, the length field in the delimiter may also be corrupt. If the delimiters CRC fails, the receiver will search forward on every 32-bit (4 byte) boundary looking for data that looks like a delimiter. There is supposedly a good chance of the receiver predicting the next delimiter [134]. Variable lengths of padding are added to the tail of frames to ensure that every delimiter starts on a 4 byte boundary. These mechanisms ensure reliability within A-MPDUs, or simply, that the loss of one packet within a A-MPDU does not result in the loss of others.

Both A-MSDU and A-MPDU can increase throughput in 802.11n networks, however, in the case of A-MPDU, I question whether there may be a more efficient solution to the problem of undersized packets. When aggregating frames, overhead is added and TCP headers, IP headers and MAC headers are often replicated. Therefore it is debaTable whether the problem of small packets may be better addressed at a higher layer.

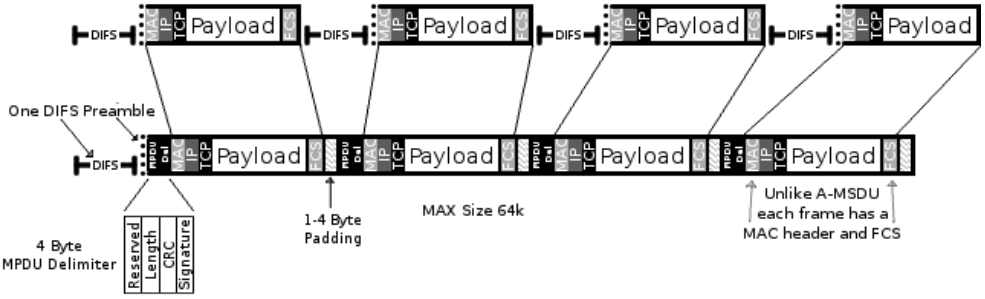


Figure B.2: A-MPDU (Aggregate MAC Protocol Data Unit)

B.3 Efficiency through MTU scaling

B.3.1 MTUs in wired and wireless networks

There are strong arguments that the 1500 byte MTU (Maximum Transmission Unit) is too small for modern communications networks [135, 136, 137]. Despite the large speed increases, mandatory Ethernet packet sizes have remained static since 1982[1]. Table B.1 shows that as Ethernet speeds have increased, serialization delays have dramatically decreased and MTUs have remained stagnant.

Small MTU's are inefficient for a few reasons. The packet headers contribute to an unnecessarily high percentage of overhead. Furthermore, there is also evidence suggesting that many small frames cause more CPU interrupts and processing overhead [137]. Whether this is still true, given increased CPU speeds and perhaps the migration of functionally onto NIC's is unknown. The original argument against using larger packet sizes in wired networks was serialization delays. However, Table B.1 clearly shows that in

[1] Jumbo frames are allowed but are not mandatory

Table B.1: Ethernet evolution and mandatory MTU's

Technology	Rate	Year	MTU	Serialization Delay
Ethernet	10Mb/s	1982	1500	1200μs
Fast Ethernet	100Mb/s	1995	1500	120 μs
Gigabit Ethernet	1000Mb/s	1998	1500	12 μs
10 Gigabit Ethernet	10000Mb/s	2002	1500	1.2 μs

Table B.2: 802.11 evolution and mandatory MTUs

Technology	Rate	Year	MTU	1500b s-delay	Max MTU s-delay
802.11b	11Mb/s	1999	2272	1091μs	1653μs
802.11g	54Mb/s	2003	2272	222μs	337μs
802.11n	432Mb/s	2009-2010	7935	28μs	147μs

modern Ethernet networks, this argument is no longer valid.

Wireless 802.11 networks have higher Mandatory MTU's than 802.3 Ethernet networks. Table B.2 shows the MTU's for the different 802.11 physical layers. Note that the Jump in MTU for 802.11n was likely made to accommodate A-MSDU. Similar to Ethernet networks, using modern wireless technologies, serialization delays are no longer a limiting factor. The maximum frame size of 7935 bytes in 802.11n is now serialized in about 7.5 times less than a 1500 byte frame in 802.11b.

Transmitting larger packets rather than aggregating 1500 byte packets has a number of benefits. The most obvious is the savings on, MAC headers, IP headers and TCP headers, making transmission more efficient. In addition to this, by increasing the packet size, there are fewer packets transmitted per second, significantly reducing the number of preambles, IFS, frame trailers and 802.11 acknowledgments.

B.3.2 MTU application failure

There are a number of reasons why Internet MTU's have not scaled in the manner that would provide optimal performance. The traditional argument against larger packet sizes is that the larger serialization delays can cause high priority packets to be delayed. LAN switch vendors also found it unfavorable to do fragmentation between switches not capable of higher MTU's, requiring additional memory and processing. Perhaps another reason is because in non-supercomputing environments, the performance benefits did not justify the additional complexity. The biggest reason is because the initial MTU Discovery defined in RFC 1191 [138] was flawed in many ways. The reasons for this failure are documented in RFC 2923 [139].

In the current day, many of these traditional arguments are invalid. Wired and wireless speeds have changed such that the 1500 byte MTU used since 1982, is a significant limitation. Finally, the problems that plagued previous path MTU discovery [138] were solved in 2007 with RFC 4281 [140]. Given that MTU problems have now been rectified, an analysis of the potential benefits to wireless networks is justified. The main arguments for larger packet sizes, namely the reduction in preambles, IFS, frame trailers and 802.11 acknowledgments are obvious and have already been mentioned. Other implications of MTU scaling are more complex and deserve detailed analysis.

B.3.3 MTU's, packet loss and end-to-end bandwidth

Increasing packet sizes beyond 1500 bytes can have a significant effect on end-to-end throughput. In Mathis seminal paper [141], it is stated that end-to-end throughputs are bound by equation B.1 where, p is the loss probability:

$$BW < (\frac{MSS}{RTT})(\frac{1}{\sqrt{p}}) \tag{B.1}$$

This equation states that the effective end-to-end bandwidth, or throughput is proportional to the MSS. A simplistic analysis of this equation might be that throughputs can be doubled by doubling the MTU, however, this assumption is unfair, because for wireless 802.11 networks, loss probability (p) increases with packet sizes. The diagram in Figure B.3 demonstrates why large packets led to higher packet loss in wireless networks. It suggests that doubling the packet size will double the packet loss rate. Due to the burst nature of wireless losses, packet losses would may not increase to this extent.

If as a worst case, packet losses do increase linearly with packet sizes, end-to-end bandwidths still increase. Figure B.4 shows the effect of increasing MSS and packet losses over a range of delays. The points start with a 1500 byte packet with 2% packet loss and finishing with a 12,000 byte packet with 16% loss. While packet loss can increase with packet size, it has an inverse square effect on throughput, which means that MSS still dominates [137]. This graph shows that even if packet loss did scale linearly with packet sizes,

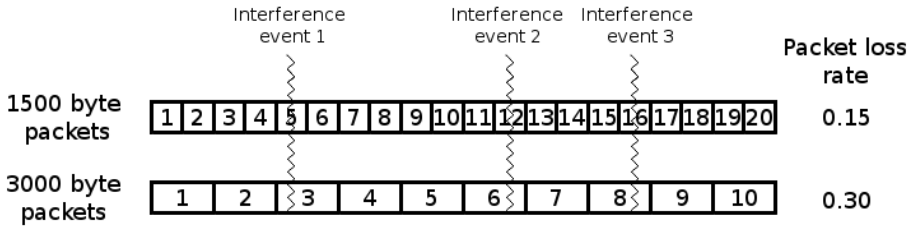


Figure B.3: Packet size and reliability

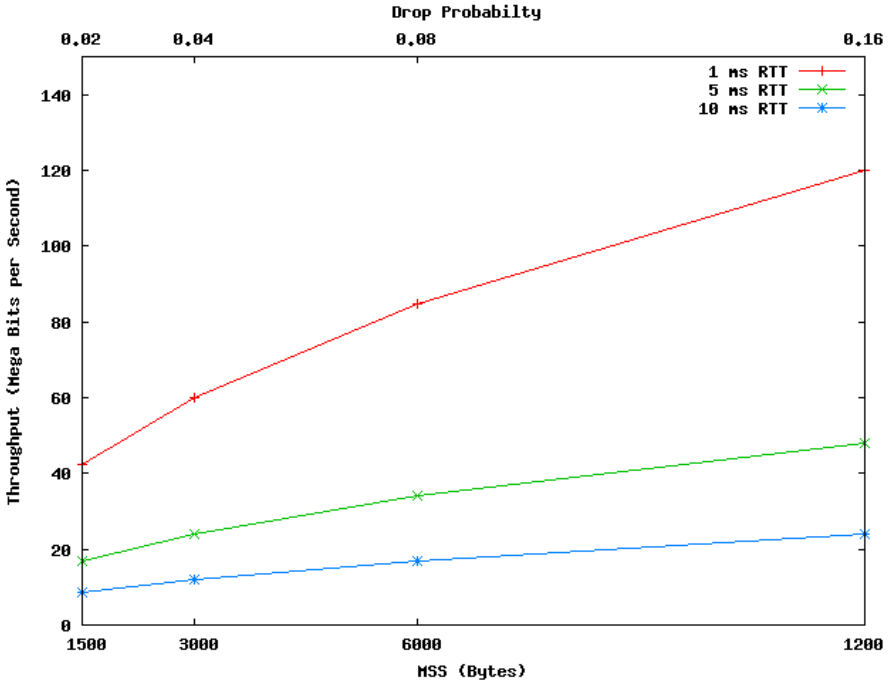


Figure B.4: End-to-end bandwidth

end to end bandwidth would still benefit from increasing packet sizes.

B.3.4 Excessive TCP acks

A further problem with small MTUs is the excessive numbers of TCP acks. With small packet sizes, the number of TCP acks increase. TCP acks are particularly inefficient in 802.11 networks for a number of reasons. As TCP acks require their own DIFS, preamble and MAC layer ack, this means that the channel time is significantly larger than the standard transmission time for a wired network. Also, the shared, half duplex nature of 802.11 means that TCP ack transmissions affect the flow of TCP data packets in the reverse direction, consuming channel time that could otherwise be used for data transmission. In many other technologies such as Ethernet or ADSL the size and number of TCP acks is less important as there are separate up and down channels. Our analysis in section 4.1.5 shows that for 802.11, when the MTU is 1500 bytes, between 15% and 17% of transmission time is consumed by the TCP ack.

We performed an experiment where to removed every fourth TCP ack (a loss rate of 25%) from a TCP transfer over 802.11. Given this very high loss rate, no difference was prevalent in the end to end transfer rate. The reason being that TCP acks are cumulative hence, if a TCP ack is lost, the arrival of the following TCP ack will acknowledge all of the previous data. Thus, increasing the MTU may generate fewer TCP acks allowing more time for data transmission. Also, using MAC layer acknowledgments to provide reliability for TCP acks is unnecessary as the experimentation suggests that high loss rates make little difference to transfer speeds.

B.4 Two problems: ack efficiency and small packets

This section began with 802.11e BlockAck efficiency improvements. The 802.11n frame aggregation feature was then discussed and contrasted with MTU scaling. Our digression into frame aggregation and MTU sizes is because the 802.11e BlockAck and 802.11n frame aggregation are often discussed as enhancements for 802.11n collectively. However, these two different efficiency schemes solve two different problems. 802.11e BlockAcks, attempt to solve the problem of acknowledgment efficiency. 802.11n frame aggregation aims to solve the problem of too many small frames.

Following the discussion of frame aggregation, MTU scaling which is a higher layer solution to the problem of too many small frames was discussed. I question whether larger MTUs are significantly better for end-to-end transfers than 802.11n frame aggregation. The reason being that, A-MSDU saves time on on IFS and preambles. Comparatively, larger MTUs reduce the number of packets transmitted, saving on on MAC headers, IP headers, TCP headers and TCP acknowledgments. These mechanisms are contrasted in Figure B.5 which shows standard 802.11 DCF, BlockAck, 802.11 Frame aggregation with BlockAcks and finally standard 802.11 DCF with larger packet sizes. This diagram is not to scale with time and does not suggest that standards based 802.11 with large MTUs is any percentage more efficient than 802.11n frame aggregation with BlockAck. Comparing those two techniques would be a complex procedure and is worthy of a separate dedicated study.

This chapter proposes to solve the acknowledgment efficiency problem. The problem of small frames may be solved in conjunction with this mechanism by raising the MTU. Evidence suggests that MTU scaling may have

many positive effects at both the link layer and the transport layer. However, I believe that the solutions to the different problems should be compared independently. The reason for the lengthy digression away from ack efficiency and into frame aggregation and MTU scaling was to inform the reader of MAC layer changes in 802.11e and 802.11n. This should provide the understanding to make fair comparisons between solutions, suggested later in this chapter, and the 802.11 BlockAck function.

The discussion of frame aggregation and MTU scaling also shows that many of the inefficiencies in modern networks can be effectively solved at many different layers. It also shows that many new data communications technologies blur the traditional layered architecture. Frame aggregation solves the inefficiencies associated with large numbers of small frames. However, it could be argued that packet size was originally intended to be mediated by MTUs. Equally, 802.11 acknowledgments don't provide end-to-end reliability, instead they prevent TCP from misinterpreting packets lost due to interference as congestion.

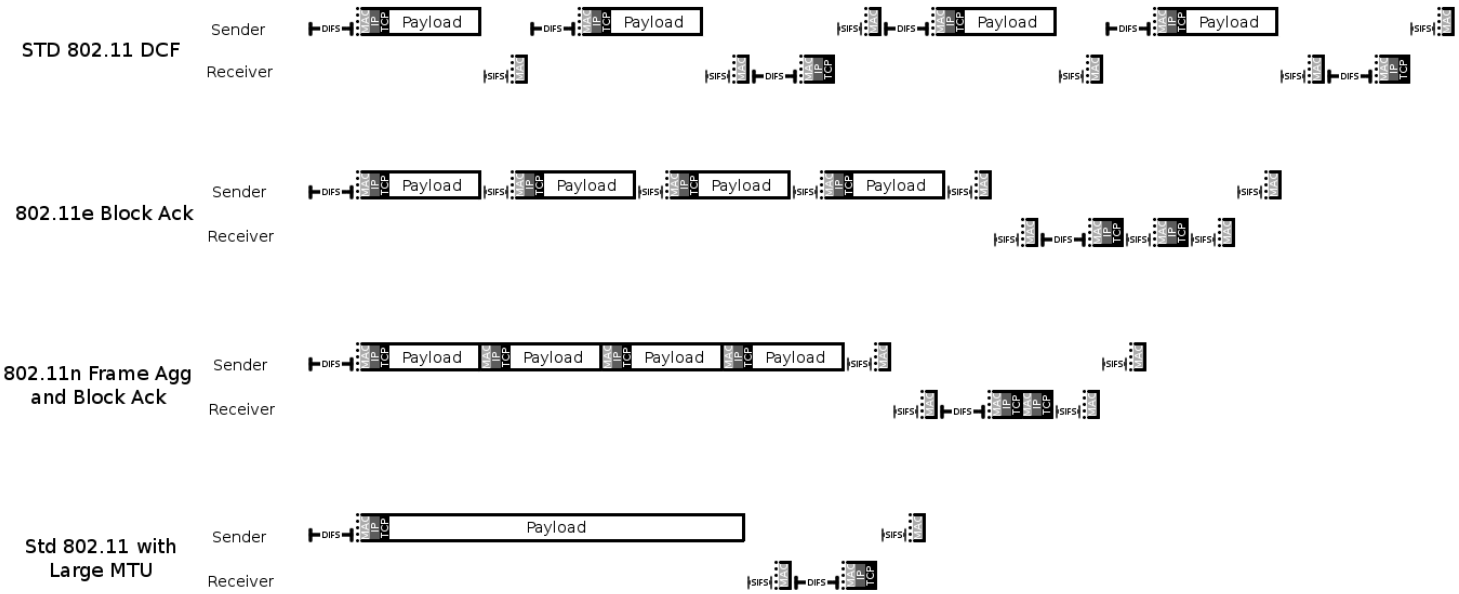


Figure B.5: 802.11 efficiency techniques

References

[1] IEEE, "802.16j - IEEE Standard for Local and metropolitan area networks - Part 16: Air Interface for Broadband Wireless Access Systems - Amendment 1: Multihop Relay Specification." IEEE Standard, June 2009.

[2] W. T. H. Woon and T.-C. Wan, "Performance evaluation of IEEE 802.15.4 wireless multihop networks: simulation and testbed approach," *International Journal of Ad Hoc Ubiquitous Computing*, vol. 3, no. 1, pp. 57–66, 2007.

[3] IEEE, "802.11 - IEEE Standards for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Network - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE standards, 2007.

[4] IETF, "Mobile Ad-hoc Networks (MANET)." http://www.ietf.org/html.charters/manet-charter.html, 2007.

[5] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations." IETF RFC 2501, January 1999.

[6] OLPC, "One Laptop Per Child." http://laptop.org/vision/index.shtml, 2007.

[7] International Telecommunication Union, "Data Network and Open System Communications." ITU standards document, July 1994.

[8] V. Kawadia and P. R. Kumar, "A Cautionary Perspective On Cross-layer Design," *IEEE Wireless Communications*, vol. 12, pp. 3–11, 2005.

[9] P. Guptak and P. R. Kumar, "The Capacity of Wireless Networks," in *IEEE Transactions on Information Theory*, 2000.

[10] P. Gupta, R. Gray, and P. R. Kumar, "An Experimental Scaling Law for Ad Hoc Networks," 2001.

[11] T. R. Andel and A. Yasinsac, "On the Credibility of MANET Simulations," *Computer*, vol. 39, pp. 48–54, July 2006.

[12] P. Hu, A. Pirzada, and M. Portmann, "Experimental evaluation of aodv in a hybrid wireless mesh network," in *The 5th Workshop on the Internet, Telecommunications and Signal Processing WITSP'06*, 2006.

[13] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *SIGCOMM Computer Communications Review*, vol. 24, no. 4, pp. 234–244, 1994.

[14] D. B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," in *In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 158–163, 1994.

[15] A. Tonnesen, "Implementing and extending the Optimized Link State Routing Protocol," Master's thesis, University of Oslo, Department of Informatics, 2004.

[16] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)." IETF RFC 3626, October 2003.

[17] T. Clausen and C. D. P. Jacquet, "The Optimized Link State Routing Protocol version 2." IETF Draft RFC draft-ietf-manet-olsrv2-10, September 2009.

[18] J. Chroboczek, "The Babel Routing Protocol." Internet-Draft, April 2009.

[19] A. Neumann, C. Aichele, and M. Lindner, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)." IETF Draft, October 2008.

[20] IEEE-TGs, "Joint SEE-Mesh / Wi-Mesh Proposal to 802.11 TGs." IEEE proposal, 2006.

[21] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot, "Performance of Multipoint Relaying in Ad Hoc Mobile Routing Protocols," in *NETWORKING '02: Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, (London, UK), pp. 387–398, Springer-Verlag, 2002.

[22] J. Roy, "OSPF Version 2." IETF RFC 2328, April 1998.

[23] B. Albrightson, J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP - A Fast Routing Protocol Based On Distance Vectors," in *Networld/Interop 94*, 1994.

[24] Andreas Tonnesen and Thomas Lopatic and Hannes Gredler and Bernd Petrovitsch and Aaron Kaplan and Sven-Ola Tucke, "OLSR development at http://www.olsr.org/." Internet: http://www.olsr.org/, 2009.

[25] J. J. Garcia-Lunes-Aceves, "Loop-Free Routing Using Diffusing Computations," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 130–141, 1993.

[26] T. R. Henderson, P. A. Spagnolo, and J. H. Kim, "A Wireless Interface Type for OSPF," *MILCOM Military Communications Conference*, vol. 2, pp. 1256 – 1261, 2003.

[27] R. Ogier and A. Spagnolo, "Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding." IETF Experimental RFC 5614, August 2009.

[28] K. Holter, "Wireless Extensions to OSPF: Implementation of the Overlapping Relays Proposal," Master's thesis, University of Oslo, Department of Informatics, 2006.

[29] P. A. Spagnolo and T. R. Henderson, "Comparison of Proposed Ospf Manet Extensions," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–7, Oct. 2006.

[30] C. A. Santivanez and R. Ramanathan, "Scalability of Routing in Ad Hoc Networks: Principles and Practice," *Kluwer: Ad Hoc Wireless Networking*, vol. 1, 2003.

[31] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4." IETF RFC 4728, February 2007.

[32] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing." IETF RFC 3561, 2003.

[33] S. R. Das, C. E. Perkins, and E. E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," in *INFOCOM (1)*, pp. 3–12, 2000.

[34] K. Seada, C. Westphal, and C. Perkins, "Analyzing Path Accumulation for Route Discovery in Ad Hoc Networks," in *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pp. 4377–4382, March 2007.

[35] I. Chakeres and C. E. Perkins, "Dynamic MANET On-demand (DYMO) Routing, Internet-Draft." IETF Mobile Ad Hoc Networks Working Group, 2007.

[36] M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1395–1414, 1999.

[37] 802.11-TGs, "Status of Project IEEE 802.11s - ESS Mesh Networking." http://grouper.ieee.org/groups/802/11/Reports/, 2006.

[38] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," in *MobiCom '98: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, (New York, NY, USA), pp. 76–84, ACM, 1998.

[39] G. P. Mario, M. Gerla, and T. wei Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks," in *in Proceedings of ICC 2000*, pp. 70–74, 2000.

[40] J. P. Macker and J. W. Dean, "A Study of Link State Flooding Optimizations For Scalable Wireless Networks," in *MILCOM 2003: Military Communications Conference*, vol. 2, pp. 1262–1267, IEEE, 2003.

[41] C. A. Santivanez and R. Ramanathan, *Ad Hoc Wireless Networking*, ch. Scalability of Routing in Ad Hoc Networks: Principles and Practice, pp. 562 – 616. Kluwer Acedemic Publishers, 2003.

[42] C. A. Santivanez and R. Ramanathan, "Hazy Sighted Link State (HSLS) Routing: A Scalable Link State Algorithm," tech. rep., Internetwork Research Department, BBN Technologies, 2001.

[43] E. M. Royer, *Routing in Ad hoc Mobile Networks: On-Demand and Hierarchial Strategies*. PhD thesis, University of California, 2000.

[44] W. Heo and M. Oh, "Performance of Expanding Ring Search Scheme in AODV Routing Algorithm," in *FGCN '08: Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking*, (Washington, DC, USA), pp. 128–132, IEEE Computer Society, 2008.

[45] C. Adjih, P. Jacquet, and L. Viennot, "Computing Connected Dominated Sets with Multipoint Relays," *Ad Hoc & Sensor Wireless Networks*, vol. 1, pp. 27–39, 2005.

[46] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks," tech. rep., INRIA, 2000.

[47] C. B. Jerome Harri and F. Filali, "OLSR and MPR: Mutual Dependences and Performances," tech. rep., Eurecom, 2005.

[48] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 134–146, ACM Press, 2003.

[49] R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Static Multi-hop Wireless Networks," in *SIGCOMM '04: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, (New York, NY, USA), pp. 133–144, ACM Press, 2004.

[50] A. Khanna and J. Zinky, "The Revised ARPANET Routing Metric," 1989.

[51] *Feasibility study of mesh networks for all-wireless offices*, (New York, NY, USA), ACM, 2006.

[52] I. F. Akyildiz and X. Wang, *Wireless Mesh Networks*. Wiley, 2009.

[53] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 114–128, ACM Press, 2004.

[54] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad hoc Wireless Networks," in *SIGCOMM*, 1996.

[55] C. Perkins and E. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1997.

[56] M. Linder and A. Neumann, "BATMAN (Better Approach To Mobile Ad Hoc Networking)." http://open-mesh.net/batman/.

[57] Gianni Di Caro and Gianni Di Caro and Frederick Ducatelle and Frederick Ducatelle and Luca Maria Gambardella and Luca Maria Gambardella, "AntHocNet: An Adaptive Nature-inspired Algorithm for Routing in Mobile Ad hoc Networks," *European Transactions on Telecommunications*, vol. 16, pp. 443–455, 2005.

[58] open80211s, "open80211s." http://www.open80211s.org/.

[59] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang., "Real-world performance of current proactive multi-hop mesh protocols," in *APCC 2009: Asia Pacific Conference on Communications*, 2009.

[60] D. Johnson, N. Ntlatlapa, and C. Aichel, "Simple pragmatic approach to mesh routing using BATMAN," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, 2008.

[61] A. Tonnesen, "OLSR." http://www.olsr.org/, 2007.

[62] J. Chroboczek, "Babel a loop-free distance-vector routing protocol." http://www.pps.jussieu.fr/ jch/software/babel/, 2009.

[63] E. Nordstrom, "AODV-UU 0.9.5." http://linux.softpedia.com/get/System/Networking/AOD
UU-15587.shtml, 2009.

[64] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *SIGMOBILE Mobile Computer Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.

[65] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," in *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, (Rome, Italy), pp. 61–69, July 2001.

[66] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "Wi-Fi in Ad Hoc Mode: A Measurement Study," in *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications*, (Washington, DC, USA), p. 145, IEEE Computer Society, 2004.

[67] P. Kyasanur and N. H. Vaidya, "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks," in *IEEE Wireless Communications and Networking Conference*, 2005.

[68] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks," in *ISPAN '00: Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '00)*, (Washington, DC, USA), p. 232, IEEE Computer Society, 2000.

[69] N. Jain, S. R. Das, and A. Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *Tenth International Conference on Computer Communications and Networks*, 2001.

[70] P. Kyasanur, J. Padhye, and P. Bahl, "On the efficacy of separating control and data into different frequency bands," *2nd International Conference on Broadband Networks*, vol. 1, pp. 602 – 611, 2006.

[71] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-hoc Wireless

Networks," in *MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, (New York, NY, USA), pp. 216–230, ACM Press, 2004.

[72] J. So and N. H. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 222–233, ACM, 2004.

[73] B. Hodge, "Timer Synchronization Function (TSF) Scaling Issues in 802.11." Online: http://smartnets.eecs.berkeley.edu/papers/SynchronizationWriteUp.doc, 2003.

[74] I. Ramani and S. Savage, "Syncscan: Practical Fast Handoff for 802.11 Infrastructure Networks," *IEEE INFOCOM The Conference on Computer Communications*, vol. 1, pp. 675 – 678, 2005.

[75] M. Shin, A. Mishra, and W. A. Arbaugh, "Improving the latency of 802.11 hand-offs using neighbor graphs," in *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, (New York, NY, USA), pp. 70–83, ACM Press, 2004.

[76] D. Murray, T. Koziniec, and M. Dixon, "An Analysis of Handoff in Multi-band 802.11 Networks," in *MASS'07: 4th IEEE Conference on Mobile Ad-Hoc and Sensor Systems*, 2007.

[77] D. Murray, "Fast and Efficient Scanning Mechanism in Wireless Local Area Networks, Honours Thesis," Master's thesis, School of IT, Murdoch University, 2007.

[78] "Linux wireless." Online: http://linuxwireless.org/en/users, 2008.

[79] Y. Yang, J. Wang, and R. Kravets, "Designing Routing Metrics for Mesh Networks," in *IEEE WiMesh 2005*, (Santa Clara, Ca), September 2005.

[80] Y. Yang, J. Wang, and R. Kravets, "Interference-aware Load Balancing for Multihop Wireless Networks," tech. rep., University of Illinois at Urbana-Champaign, 2005.

[81] A. A. Pirzada, M. Portmann, and J. Indulska, "Performance analysis of multi-radio aodv in hybrid wireless mesh networks," *Computer Communications*, vol. 31, pp. 885–895, 2007.

[82] J. Tang, G. Xue, and W. Zhang, "Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks," in *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 68–77, ACM, 2005.

[83] M. K. Marina and S. R. Das, "A Topology Control Approach for Utilizing Multiple Channels in Multi-Radio Wireless Mesh Networks," in *2nd International Conference on Broadband Networks*, vol. Vol. 1, pp. 381 – 390, 2005.

[84] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-Aware Channel Assignement in Multi-Radio Wireless Mesh Networks," in *IEEE INFOCOM*, 2006.

[85] M. Alicherry, R. Bhatia, and L. Erran, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *MobiCom '05: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, (New York, NY, USA), pp. 58–72, ACM, 2005.

[86] S. M. Das, H. Pucha, D. Koutsonikolas, C. Hu, and D. Peroulis, "DMesh: Incorporating Practical Directional Antennas in Multichannel Wireless Mesh Networks," in *IEEE Journal on Selected Areas in Communications*, 2006.

[87] R. Chandra, C. Fetzer, and K. Hogstedt, "Adaptive Topology Discovery in Hybrid Wireless Networks." Microsoft Research, 2002.

[88] N. Migas, W. J. Buchanan, and K. A. M. Aartney, "Mobile Agents for Routing, Topology Discovery, and Automatic Network Reconfiguration in Ad-Hoc Networks," in *ECBS'03: 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 2003.

[89] B.-J. Ko, V. Misra, J. Padhye, and D. Rubenstein, "Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks," in *WCNC: Wireless Communications and Networking Conference*, 2007.

[90] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *INFOCOM - 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.

[91] A. Raniwala, R. Krishnan, and T. Chiueh, *Wireless Mesh Networking: Architectures, Protocols and Standards*, ch. IEEE 802.11-Based Wireless Mesh Networks, pp. 79 – 109. Auerbach, 2007.

[92] P. Fuxjager, D. Valerio, and F. Ricciato, "The Myth of Non-Overlapping Channels: Interference Measurements in IEEE 802.11," in *WONS: Wireless on Demand Network Systems and Services*, 2007.

[93] D. Murray, M. Dixon, and T. Koziniec, "Scanning Delays in 802.11 Networks," in *NGMAST'07: International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies*, 2007.

[94] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM SIGCOMM Computer Communications Review*, vol. 33, pp. 93 –102, 2002.

[95] J.-O. Vatn, "An experimental study of ieee 802.11b handover performance and its effect on voice traffic," tech. rep., KTH Royal Institute of Technology, 2003.

[96] H. Velayos and G. Karlsson, "Techniques to reduce IEEE 802.11b MAC layer handover time," *ICC IEEE Conference on Communications*, vol. 1, pp. 3844 – 3848, 2004.

[97] J. Tourrilhes, "Wireless Tools for Linux." http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html, 2008.

[98] W. Si and S. Selvakennedy, "A Survey on Channel Assignment Approaches for Multi-Radio Multi-Channel Wireless Mesh Networks," tech. rep., University of Sydney, 2007.

[99] IEEE, "802.11a - IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications-Amendment 1: High-speed Physical Layer in the 5 GHz band." IEEE Standards, 1999.

[100] Q. Pang, S. C. Liew, and V. C. M. Leung, "Performance Improvement of 802.11 Wireless Network with TCP ACK agent and auto-zoom backoff algorithm," in *IEEE 61st Conference on Vehicular Technology*, vol. 3, pp. 2046–2050 Vol. 3, 2005.

[101] J. Barcelo, B. Bellalta, A. Sfairopoulou, C. Cano, and M. Oliver, "No Ack in IEEE 802.11e Single-Hop Ad-Hoc VoIP Networks," in *The 7th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, 2008.

[102] IEEE, "802.11e - IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)

specifications-Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements." IEEE Standards, 2005.

[103] IEEE, "802.11n - 2009 Standard for Information technology - Telecommunications and information exchange between systems - Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) & Physical Layer specifications Enhancements for Higher Throughput." IEEE standards, October 2009.

[104] O. Cabral, A. Segarra, and F. J. Velez, "Implementation of Multiservice IEEE 802.11e Block Acknowledgement Policies," *IAENG International Journal of Computer Science*, vol. 36:1, p. 1, June 2009.

[105] O. Cabral, A. Segarra, and F. J. Velez, "Implementation of IEEE 802.11e Block Acknowledgement Policies Based on the Buffer Size," in *14th European Wireless Conference*, pp. 1–7, June 2008.

[106] G. R. Hiertz, L. Stibor, J. Habetha, E. Weiss, and S. Mangold, "Throughput and Delay Performance of IEEE 802.11e Wireless LAN with Block Acknowledgments," in *11th European Wireless Conference*, 2005.

[107] T. Li, Q. Ni, and Y. Xiao, "Investigation of The Block ACK Scheme in Wireless Ad hoc Networks: Research Articles," *Wireless Communications & Mobile Computing*, vol. 6, no. 6, pp. 877–888, 2006.

[108] T. Li, Q. Ni, T. Turletti, and Y. Xiao, "Performance Analysis of the

IEEE 802.11e Block ACK Scheme in a Noisy Channel," in *IEEE BroadNets*, pp. 551–557, 2005.

[109] A. Ranjitkar and Y.-B. Ko, "Performance Enhancement of IEEE 802.11s Mesh Networks using Aggressive Block Ack Scheme," in *The International Conference on Information Networking*, 2008.

[110] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic," in *ACM Computer Communication Review*, pp. 133–147, 1991.

[111] H. Balakrishnan, V. N. Padmanabhan, G. Fairhurst, and M. Sooriyabandara, "TCP Performance Implications of Network Path Asymmetry." IETF RFC 3449, December 2002.

[112] D. X. Wei, P. Cao, and S. H. Low, "TCP Pacing Revisted," in *Infocom*, 2006.

[113] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," in *INFOCOM*, vol. 3, pp. 1157–1165 vol.3, Mar 2000.

[114] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 588–603, 2005.

[115] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options." IETF FRC 2018, October 1996.

[116] F. Anjum and L. Tassiulas, "Comparative study of various TCP versions over a wireless link with correlated losses," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 370–383, 2003.

[117] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "TCP Westwood: Congestion Window Control Using Bandwidth Estimation," in *IEEE Globecom*, vol. 3, pp. 1698–1702, November 2001.

[118] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," in *ACM Mobicom*, pp. 287–297, July 2001.

[119] W. Ding and A. Jamalipour, "A new explicit loss notification with acknowledgment for wireless TCP," in *2001 12th IEEE International Symposium onPersonal, Indoor and Mobile Radio Communications,*, vol. 1, pp. B–65–B–69 vol.1, Sep 2001.

[120] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[121] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations." IETF RFC 3135, June 2001.

[122] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," in *15th International Conference on Distributed Computing Systems*, pp. 136–143, 1995.

[123] M. Luglio, M. Y. Sanadidi, M. Gerla, and J. Stepanek, "On-board Satellite Split TCP Proxy," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 362–370, February 2004.

[124] C. Caini, R. Firrincieli, and D. Lacamera, "PEPsal: A Performance Enhancing Proxy for TCP Satellite Connections," in *IEEE Aerospace and Electronic Systems, Internetworking and Resource Management in Satellite Systems Series*, vol. 22, pp. B–9–B–16, Aug. 2007.

[125] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over Wireless Networks," in *1st Annual International Conference on Mobile Computing and Networking*, (New York, NY, USA), pp. 2–11, ACM, 1995.

[126] H. Balakrishnan and R. H. Katz, "Explicit Loss Notification and Wireless Web Performance," in *IEEE Globecom Internet Mini-Conference*, 1998.

[127] C. H. Ng, J. Chow, and L. Trajkovic, "Performance Evaluation of TCP over WLAN 802.11 with the Snoop Performance Enhancing Proxy," in *In Proceedings of OPNETWORK*, pp. 134–142, 2002.

[128] F. Sun, V. O. Li, and S. C. Liew., "Design of SNACK Mechanism for Wireless TCP with New Snoop," in *Wireless Communications and Networking Conference*, vol. 2, pp. 1051–1056, March 2004.

[129] J. Kim and K. Chung, "C-Snoop: Cross Layer Approach to Improving TCP Performance Over Wired and Wireless Networks," *International*

Journal of Computer Science and Network Security, vol. 7(3), pp. 131–137, 2007.

[130] S. Vangala and M. A. Labrador, "The TCP SACK-Aware Snoop Protocol for TCP over Wireless Networks," in *IEEE Vehicular Technology Conference*, 2002.

[131] Freifunk, "Freifunk Wiki." http://freifunk.net/wiki/FreifunkFirmwareEnglish, 2007.

[132] OpenWRT, "OpenWRT - Wireless Freedom." http://openwrt.org/, 2008.

[133] J. Tourrilhes, "Packet frame grouping: improving IP multimedia performance over CSMA/CA," in *IEEE International Conference on Universal Personal Communications*, vol. 2, pp. 1345–1349 vol.2, Oct 1998.

[134] E. Perahia and R. Stacey, *Next Generation Wireless LANs : Throughput, Robustness, and Reliability in 802.11n.* Cambridge University Press, 2008.

[135] M. Mathis, "Raising the Internet MTU." http://staff.psc.edu/mathis/MTU/, 2009.

[136] W. Rutherford, L. Jorgenson, M. Siegert, P. V. Epp, and L. Liu, "16 000-64 000 B pMTU experiments with simulation: The case for super jumbo frames," *Supercomputing: Optical Switching and Networking*, vol. 4(2), pp. 121–130, 2005.

[137] P. Dykstra, "Gigabit Ethernet Jumbo Frames and Why You Should Care." www.wareonearth.com/whitepapers/GigabitEthernetJumboFrames.pdf, December 1999.

[138] J. Mogul and S. Deering, "Path MTU Discovery, RFC 1191." IETF RFC, 1990.

[139] K. Lahey, "TCP Problems with Path MTU Discovery, IETF RFC 2923." IETF RFC, 2000.

[140] M. Mathis and J. Heffner, "Packetization Layer Path MTU Discovery, RFC 4821." IETF RFC, 2007.

[141] M. Mathis, J. Semke, and J. Mahdavi, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *Computer Communications Review*, vol. 27(3), no. 3, pp. 67–82, 1997.